Explorando conceitos da teoria de espaços métricos em consultas por similaridade sobre dados complexos[1]

Ives Renê Venturini Pola
ives@icmc.usp.br

Orientador:
Prof. Dr. Caetano Traina Júnior
caetano@icmc.usp.br

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC, USP, como parte dos requisitos para a obtenção do título de Doutor em Ciências de Computação e Matemática Computacional.

USP - São Carlos
Junho de 2010

[1]Este trabalho tem o apoio financeiro da **CAPES**, **FAPESP** e **CNPq**.

Explorando conceitos da teoria de espaços métricos em consultas por similaridade sobre dados complexos

Ives Renê Venturini Pola

Publish or perish
Demo or die

Agradecimentos

Ao meu orientador e amigo, Prof. Dr. Caetano Traina Júnior, agradeço o incentivo, a confiança e a dedicação a mim dados durante todo o período de elaboração deste trabalho. Além de um grande orientador, um grande amigo e professor, pois me apoiou quando precisei e me ensinou muito, tanto no aspecto pessoal quanto no profissional.

À Fernanda, o amor de minha vida, minha companheira eterna, agradeço a paciência, o carinho e a colaboração feita neste trabalho. Sem você minha vida seria menos feliz, e eu viveria incompleto no meu ser, pois você é minha alma gêmea.

Aos meus pais, Izes e Osmar, por acreditar em mim, por dar a oportunidade e suporte aos meus estudos, pelo incentivo e pela compreensão nesta grande etapa em minha vida. À minha irmã Isis por me apoiar sempre e me incentivar. Eu os amo muito, obrigado por me ajudar quando precisei, e pela paciência tida quando estive ausente.

À Profa. Dra. Agma J. M. Traina, pelos ensinamentos e orientações, pela maneira que sempre me tratou, realmente uma grande amiga e uma excelente companheira profissional.

Aos colegas do grupo de bases de dados, GBDI, com os quais compartilhei experiência e aprendi bastante em diversas linhas de pesquisa.

A todos os funcionários do ICMC, que sempre me prestaram palavras de encorajamento neste etapa de meus estudos.

À CNPq, CAPES e FAPESP, pelo suporte financeiro em diversas modalidades.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Siglas e Símbolos

SGBD	Sistema de Gerenciamento de Banco de Dados
MAM	Método de acesso métrico
MAE	Método de acesso espacial
ISAM	*Indexed Sequential Access Method*
CBIR	*Content Based Image Retrieval*
SIG	Sistema de Informação Geográfica
$\mathbb{S}$	Domínio de dados que segue uma determinada formação
S	Conjunto de elementos que seguem a mesma formação definida em um domínio, $S \subseteq \mathbb{S}$.
s_i	Elemento de S, ou seja, $s_i \in S$
$\mathbb{R}^n$	Espaço Vetorial com cardinalidade n
$a = \{a_1, a_2, ..., a_n\}$	Elemento de um espaço vetorial n-dimensional
a_i	Valor do elemento a na dimensão i
$d : \mathbb{S}$ x $\mathbb{S} \rightarrow \mathbb{R}^+$	Métrica (algumas vezes chamada de função de distância, na área da computação)
$\mathbb{M} < \mathbb{S}, d >$	Domínio métrico definido sobre o domínio $\mathbb{S}$ usando a métrica d
$M < S, d >$	Espaço métrico definido sobre o conjunto $\mathbb{S}$ usando a métrica d, $M \subseteq \mathbb{M}$
$kNNq$	Consulta aos k vizinhos mais próximos
Rq	Consulta por abrangência

Resumo

Estruturas de indexação para domínios métricos são úteis para agilizar consultas por similaridade sobre dados complexos, tais como imagens, onde o custo computacional da comparação de dois itens de dados geralmente é alto. O estado da arte para executar consultas por similaridade está centrado na utilização dos chamados "Métodos de Acesso Métrico" (MAM). Tais métodos consideram os dados como elementos de um espaço métrico, onde apenas valem as propriedades fundamentais para que um espaço seja considerado métrico, onde a única informação que os MAMs utilizam é a medida de similaridade entre pares de elementos do domínio. No campo teórico, espaços métricos são extensamente estudados e servem de base para diversas áreas da Matemática. No entanto, a maioria dos trabalhos que têm sido desenvolvidos em Computação se restringem a utilizar as definições básicas desses espaços, e não foram encontrados estudos que explorem em mais profundidade os muitos conceitos teóricos existentes. Assim, este trabalho aplica conceitos teóricos importantes da Teoria de Espaços Métricos para desenvolver técnicas que auxiliem o tratamento e a manipulação dos diversos dados complexos, visando principalmente o desenvolvimento de métodos de indexação mais eficientes. É desenvolvida uma técnica para realizar um mapeamento de espaços métricos que leva à atenuação do efeito da maldição da dimensionalidade, a partir de uma aplicação lipschitziana real baseada em uma função de deformação do espaço das distâncias entre os elementos do conjunto. Foi mostrado que uma função do tipo exponecial deforma as distâncias de modo a diminuir os efeitos da maldição da dimensionalidade, melhorando assim o desempenho nas consultas. Uma segunda contribuição é o desenvolvimento de uma técnica para a imersão de espaços métricos, realizada de maneira a preservar a ordem das distâncias, possibilitando a utilização de propriedades no espaço de imersão. A imersão de espaços métricos no $\mathbb{R}^n$ possibilita a utilização da lei dos cossenos e assim viabiliza o cálculo de distâncias entre elementos e um hiperplano métrico, permitindo aumentar a agilidade à consultas por similaridade. O uso do hiperplano métrico foi exemplificado construindo uma árvore binária métrica, e também foi aplicado em um método de acesso métrico, a família MMH de métodos de acesso métrico, melhorando o particionamento do espaço dos dados.

Abstract

The access methods designed for metric domains are useful to answer similarity queries on any type of data, being specially useful to index complex data, such as images, where the computacional cost of comparison are high. The main mecanism used up to now to perform similarity queries is centered on "Metric Acess Methods" (MAM). Such methods consider data as elements that belong to a metric space, where only hold the properties that define the metric space. Therefore, the only information that a MAM can use is the similarity measure between pairs of elements in the domain. Metric spaces are extremelly well studied and is the basis for many mathematics areas. However, most researches from computer science are restrained to use the basic properties of metric spaces, not exploring the various existing theorical concepts. This work apply theoretical concepts of metric spaces to develop techniques aiding the treatment and manipulation of diverse complex data, aiming at developing more efficient indexing methods. A technique of mapping spaces was developed in order to ease the dimensionality curse effects, basing on a real lipschitz application that uses a stretching function that changes the distance space of elements. It was shown that an exponential function changes the distances space reducing the dimensionality curse effects, improving query operations. A second contribution is the developing of a technique based on metric space immersion, preserving the distances order between pairs of elements, allowing the usage of immersion space properties. The immersion of metric spaces into $\mathbb{R}^n$ allow the usage of the cossine law leading to the determination of distances between elements and a hiperplane, forming metric hiperplanes. The use of the metric hiperplanes lead to an improvement of query operations performance. The metric hiperplane itself formed the binary metric tree, and when applied to a metric access method, lead the formation of a family of metric access methods that improves the metric space particioning achieving faster similarity queries.

Capítulo

1

Introdução

1.1 Considerações gerais

Os Sistemas de Gerenciamento de Bases de Dados (SGBD) Relacionais e os desenvolvimentos mais recentes voltados aos SGBD Objeto-Relacionais foram todos inicialmente projetados para manipular apenas dados de tipos numéricos ou de pequenas cadeias de caracteres. Sobre esses dados existem fundamentalmente dois tipos de operadores de comparação que são amplamente utilizados em SGBD: os operadores para consultas por igualdade e os operadores relacionais. Os operadores de consulta por igualdade podem ser universalmente aplicados a qualquer tipo de dado, pois é sempre possível decidir se dois elementos de dado são iguais ou não. Os operadores relacionais necessitam que os dados comparados estejam em domínios que atendam à chamada relação de ordem total. Esta propriedade permite comparar qualquer par de elementos e decidir qual deles precede e qual sucede o outro ou, de maneira equivalente, qual é o maior ou menor dos dois. Juntos, os operadores relacionais e por igualdade compõem os seis operadores comumente encontrados em SGBD tradicionais, chamados "the big six" da linguagem SQL: $=, \neq, <, >, \leq e \geq$ [Melton & Simon, 2001]. Esses operadores são aplicáveis a todos os tipos de dados tradicionais dos SGBD, basicamente números, pequenas cadeias de caracteres, datas e períodos de tempo. Outros operadores de comparação, tradicionalmente suportados em SGBD tradicionais, são aplicáveis a apenas alguns dos tipos de dados tradicionais, tais como os operadores de sobreposição, que são

aplicáveis apenas a cadeias de caracteres e a períodos de tempo.

A aplicabilidade dos operadores relacionais e de igualdade a todos os tipos de dados tradicionais em SGBD levou ao desenvolvimento tanto de técnicas de indexação muito eficientes, como as *B-trees* [Comer, 1979] e suas variantes, quanto de estruturas de representação sintática uniforme para essas consultas nas linguagens de consulta, para todos os tipos de dados tradicionais. No entanto, os requisitos impostos por muitas das aplicações mais novas sobre os SGBD têm levado à necessidade de que se tratem tanto outros tipos de dados quanto outros tipos de consultas que sejam adequadas a eles. Um exemplo de tipo de dado que requer tipos de consultas específicas são os espaciais, como por exemplo Sistemas de Informações Geográficas (SIG) [Kang et al., 2009] [D'Ulizia et al., 2007] e Sistemas de Apoio a Projetos em Engenharia (CAD, do inglês *Computer Aided Design*) [Wang et al., 2010] [Ribeiro et al., 2009]. Nesses sistemas são necessários operadores de comparação específicos que envolvem a noção de espaço, tais como as consultas topológicas (como por exemplo intercepta, adjacente a, etc.) e as cardinais (aquelas baseadas em ângulos, como por exemplo ao norte, a sudeste, acima, a esquerda, etc.) [Gaede & Günther, 1998]. Modelos de custo para esta classe de estruturas são objeto de estudo no trabalho de Yu [Yu et al., 2007] e trabalhos sobre os avanços na arquitetura e tratamento de bases espaço-temporais [Medeiros et al., 2005] [Ferreira et al., 2005].

Com o advento de novas aplicações que vêm cada vez mais utilizando SGBD, vários novos tipos de dados precisam de apoio. Como a maior parte deles é muito mais elaborada do que números e pequenas cadeias de caracteres (frequentemente incorporando estruturas constituídas desses elementos), esses novos tipos têm sido chamados tipos de dados complexos. Exemplos de tipos de dados complexos são dados multimídia (imagem, áudio, vídeo, texto longo)[Leung et al., 2008][Valle et al., 2008], séries temporais [Euachongprasit & Ratanamahatana, 2008] [Montani et al., 2009] [Lian et al., 2009], informações geográficas [Kang et al., 2009], sequências de proteínas e DNA (Ácido Desoxirribonucléico) [D'Yachkov & Voronina, 2009] [Do & Wang, 2009].

Um tipo de operador de consulta importante, que se aplica de maneira geral a muitos dos tipos de dados complexos, são os operadores de consulta por similaridade [Barioni et al., 2009]. Para que eles sejam aplicáveis a um determinado domínio de dados

é necessário que seja definida uma função de distância que meça a similaridade entre pares de elementos e retorne um valor que seja tanto maior quanto mais distante um elemento estiver do outro. É comum que funções de distância sejam associadas a diversos tipos de dados, como por exemplo tipos de dados espaciais, o que habilita que se solicitem também consultas por similaridade sobre estes tipos de dados.

Os operadores relacionais não são genericamente aplicáveis a dados complexos. Por exemplo, não é possível ordenar imagens diretamente, a menos que o operador de comparação seja associado a algum atributo extra, não complexo, associado às mesmas (como por exemplo: nome, data, etc.) ou a determinadas características que podem ser extraídas dos dados complexos, estas sim ordenáveis. Mesmo a comparação por igualdade tem pouca utilidade para a maioria dos dados complexos, uma vez que a existência de dois elementos complexos exatamente iguais é raríssima [Faloutsos, 1996]. Para estes dados, o grau de similaridade é o fator mais importante, e uma vez definida uma função de distância adequada, os operadores de comparação por similaridade são muito úteis [Zhang & Zhang, 2008]. Por exemplo, SIGs usualmente adotam a função de distância euclidiana. Com a necessidade emergente do suporte a dados multimídia em SGBD, os operadores por similaridade vêm despertando interesse, principalmente para a recuperação por conteúdo de dados complexos [Datta et al., 2008].

Os dois tipos mais comuns de operadores de seleção por similaridade são: consulta por abrangência (*range query*: Rq) e consulta aos k-vizinhos mais próximos (k-nearest neighbor query: $kNNq$). Uma consulta por abrangência $Rq(s_q, \xi)$ recebe como parâmetros um elemento s_q do domínio de dados (chamado elemento central da consulta) e um limite máximo de dissimilaridade ξ e obtém todos os elementos da base de dados que diferem do elemento central da consulta por no máximo a dissimilaridade indicada. Um exemplo de consulta por abrangência sobre um conjunto de imagens é "Selecione as imagens que difiram da imagem I em no máximo 10 unidades", sendo que a unidade de distância depende da maneira como a função de dissimilaridade foi definida. As consultas por similaridade são explicadas com mais detalhes em 2.7.

Já uma consulta aos k-vizinhos mais próximos $kNNq(s_q, k)$ recebe como parâmetros um elemento s_q do domínio de dados e o número k de vizinhos desejados e obtém como resposta

os k elementos da base de dados que são os mais similares ao elemento central da consulta. Um exemplo de consulta $kNNq$ sobre um conjunto de imagens é "Selecione as 10 imagens mais semelhantes à imagem I".

As consultas por similaridade usualmente retornam o identificador dos elementos mais semelhantes ao elemento de referência da consulta. Como as operações de comparação em dados complexos são usualmente muito caras, em geral se extraem as características que descrevem adequadamente cada elemento. As operações de comparação entre elementos complexos são feitas através de comparações nas características extraídas dos elementos. Por exemplo, em imagens, podem-se utilizar características de forma, textura ou cor. Em aplicações envolvendo imagens médicas, consultas por imagens semelhantes em exames armazenados são interessantes para a localização de casos semelhantes anteriores, que são úteis para o diagnóstico de patologias e recuperação de tratamentos que se mostraram adequados.

Um **Método de Acesso** (MA) usualmente utiliza uma estrutura de dados para realizar buscas por elementos de modo eficiente, de modo que não seja necessário percorrer todos os elementos para realizar uma consulta aos dados. Ele é um dos principais recursos dos SGBDs para um maior desempenho na recuperação dos dados armazenados. O conceito fundamental que embasa um MA é o uso de determinadas propriedades existentes nos elementos indexados. Através dessas propriedades, um MA consegue descartar um subconjunto dos dados indexados sem comparar todos, mas somente alguns dos elementos indexados com o elemento de referência da consulta. Por exemplo, considere dados numéricos mantidos de maneira ordenada: neste caso a propriedade utilizada é a relação de ordem total entre os números (o número 5 é menor que 7, que por sua vez é menor que 10). Neste caso qualquer número indexado separa o conjunto em duas partes: os que são menores e os que são maiores a ele. Quando se deseja procurar por um determinado número, compará-lo com qualquer número indexado permite descartar uma das partes que o número indexado define.

Quando utilizados em SGBDs, os MAs armazenam os nós como registros em disco, todos com um tamanho fixo pré-determinado. Isso faz com que a capacidade de cada nó também seja pré-determinada e todos os nós internos e o nó raiz tenham a mesma capacidade. Os nós folha também possuem uma mesma capacidade, que pode ou não ser igual àquela dos nós internos. Armazenar os nós em registros em disco é importante, pois permite a persistência

e o suporte a um grande número de objetos. Porém, o acesso aos registros se torna mais lento, obrigando os MAs a tentar reduzir ao máximo o número de acessos a disco para responder cada consulta. Caso uma árvore tenha muitos níveis, são necessários mais acessos a disco para percorrê-la, portanto é muito importante mantê-la o menos profunda possível. Uma árvore não balanceada leva a diferentes quantidades de acessos a disco, dependendo de quais subárvores precisem ser avaliadas. Quando se permite que uma árvore seja não balanceada, é possível que seu não balanceamento a degenere completamente, pois em geral não se tem controle sobre como, ou quanto, a árvore fica desbalanceada. Por isso, apenas árvores balanceadas têm sido amplamente empregadas em SGBDs.

Um espaço métrico (mais detalhado no Capítulo 3) é definido por $\mathbb{M} =< \mathbb{S}, d >$, onde $\mathbb{S}$ é um domínio de elementos possíveis e d é a função de distância, definida para os elementos imersos em $\mathbb{S}$. Seja um subconjunto $S \subseteq \mathbb{S}$ representando um conjunto de elementos nos quais as consultas são efetuadas, a função $d : \mathbb{S}$ x $\mathbb{S} \rightarrow \mathbb{R}^+$ indica a distância entre dois elementos do domínio, onde quanto menor esta distância mais semelhantes eles são e vice-versa. O par $< \mathbb{S}, d >$ é chamado um domínio métrico sempre que a função d atender às seguintes propriedades, onde $s_1, s_2, s_3 \in \mathbb{S}$:

1. Identidade: $d(s_1, s_2) = 0 \rightarrow s_1 = s_2$
2. Simetria: $d(s_1, s_2) = d(s_2, s_1)$;
3. Não negatividade: $0 < d(s_1, s_2) < \infty$, para $s_1 \neq s_2$;
4. Desigualdade triangular: $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2))$.

A sobreposição de nós ocorre tanto em MAEs quanto em MAMs. Ela acontece quando um ou mais elementos estão coberto por duas ou mais regiões do mesmo nó, mas cada elemento está armazenado em apenas uma delas. Desta maneira, consultas nestes nós tendem a visitar todas regiões nas quais existam sobreposição na consulta. Em **árvores métricas** (MAM baseado em estrutura de dados hierárquica), o nível de sobreposição dos nós é tão ou mais influente no número de acessos a disco do que a altura. As consultas em árvores métricas têm custo diferente das demais árvores, mesmo que elas sejam balanceadas, pois pode ser necessário avaliar mais do que uma subárvore de cada nó. Neste caso, a minimização

do número de subárvores que não podem ser descartadas em uma consulta pode ser mais importante do que manter a árvore balanceada [Vieira et al., 2004].

Um problema muito importante que vem sendo pesquisado no suporte a consultas por similaridade em dados complexos é a criação de Métodos de Acesso Métrico eficientes. Um método de acesso depende dos operadores de comparação disponíveis para os dados indexados, e portanto estão vinculados a esse domínio. Os métodos de consulta para dados que atendem à relação de ordem total são muito eficientes. Eles apresentam complexidade sublinear para o número N de elementos indexados, ou seja, existem algoritmos de consulta com complexidade menor que $O(N)$. Alguns desses algoritmos recaem sobre o uso de métodos de acesso sequenciais indexados (do inglês *Indexed Sequential Access Methods* - ISAM), tais como a amplamente utilizada *B-tree* [Bayer & McCreight, 2002] e suas variantes *B*-tree* [Knuth, 1998] e *B+-tree* [Comer, 1979]. Trabalhos atuais nestas estruturas buscam maneiras eficientes de realizar o controle de concorrência [Yoshihara et al., 2008], compressão dos índices [Lin, 2008] e implementações para memórias *flash* [Roh et al., 2009] [Gal & Toledo, 2005].

Usando estes métodos, a busca por números ou pequenas cadeias de caracteres utilizando estruturas de indexação pode ser executada por algoritmos de complexidade $O(log(N))$. Estes algoritmos invariavelmente utilizam a possibilidade de comparar os elementos dois a dois, e decidir pela verdade ou falsidade de determinada relação de ordem entre cada par de elementos, o que permite evitar a comparação com ramos inteiros da árvore. Nenhum desses métodos pode ser utilizado para dados em espaços métricos, obrigando o desenvolvimento de MAMs eficientes para viabilizar o suporte a consultas por similaridade em SGBDs.

A eficiência de um método de acesso tende a aumentar se mais propriedades do domínio de dados puderem ser utilizadas. O problema é que estruturas que utilizem propriedades dos dados adicionais àquelas consideradas básicas limitam a aplicabilidade das estruturas apenas aos domínios de dados que apresentem tais propriedades adicionais. Por exemplo, a existência de domínios de dados que apresentam a propriedade de ordem total e também a propriedade de sobreposição de prefixos levou ao desenvolvimento da estrutura *String B-tree* [Ferragina & Grossi, 1999], que é muito eficiente para responder a consultas em textos, mas não pode ser utilizada por exemplo para indexar números. O interesse sobre uma deter-

minada estrutura não diminui se o conjunto de domínios de dados aos quais ela se aplica continuar amplo, embora mais restrito. Uma estrutura como a *String B-tree* é de grande interesse, mesmo que sua aplicabilidade a diferentes domínios de dados seja menor do que a de uma *B-tree*, pois grande parte dos dados usualmente armazenados numa base de dados podem ser indexados por uma *String B-tree*. De fato, várias estruturas vem sendo desenvolvidas nesta linha, principalmente na indexação de sequências genéticas [Gharib, 2009] [Stojmirović & Pestov, 2007] [Hoksza, 2009].

A maioria dos MAM utilizam poucas propriedades dos espaços métricos para agilizar os algoritmos de busca - apenas a distância entre pares de elementos, que atendem às propriedades de identidade, simetria, não-negatividade e desigualdade triangular são utilizadas. Essas propriedades têm sido fundamentais para desenvolver algoritmos que conseguem diminuir consideravelmente o tempo de execução das operações de busca, mas muitos problemas subsistem, como por exemplo a elevada sobreposição entre nós [Vieira et al., 2004]. Determinar propriedades adicionais em conjuntos de dados variados pode ser interessante para auxiliar a reduzir esses problemas e, dependendo de serem amplamente aplicáveis a uma vasta quantidade de domínios de dados diferentes, podem ser uma contribuição importante para agilizar a execução de operações de busca em conjuntos de dados complexos.

Este trabalho visa explorar conceitos da Teoria dos Espaços Métricos, de maneira mais aprofundada do que tem sido feita pelos pesquisadores da área de Ciência da Computação, visando identificar propriedades adicionais que determinados domínios de dados possam apresentar e que auxiliem no desenvolvimento de métodos de acesso métricos mais eficientes.

A Teoria dos Espaços Métricos fundamenta muitos ramos da Matemática e envolve conceitos que, sem dúvida, têm ampla aplicabilidade em muitas áreas da Computação. Assim, o objetivo foi explorar propriedades atendidas com frequência em conjuntos de dados sobre os quais exista grande interesse indexar e desenvolver algoritmos mais específico. Estes algoritmos, mais eficientes, usam propriedades adicionais que a Teoria dos Espaços Métricos indica como úteis para garantir a consistência dos mesmos, tais como os conceitos de aplicações lipschitzianas e a lei dos cossenos, os quais serão apresentados em capítulos subsequentes.

Dentre os objetivos atingidos por este trabalho, destacam-se o melhoramento do desempenho de consultas por similaridade usando aplicações lipschitzianas para separar melhor os

elementos que estejam distantes, atenuando o problema das distâncias entre quaisquer pares de elementos tenderem a ser muito semelhantes quando o conjunto de elementos conter elementos em altas dimensionalidades e a utilização da lei dos cossenos em espaços métricos usando imersões de espaços métricos no $\mathbb{R}^n$ para melhor dividir o espaço métrico em MAMs, melhorando assim seu desempenho na poda de subárvores. Portanto, esta tese mostra que a utilização do ferramental matemático dado pela Teoria dos Espaços Métricos pode dar auxílio para o desenvolvimento de técnicas para o tratamento de dados complexos, principalmente para projetar MAMs mais eficientes.

1.2 Organização do documento

Este documento apresenta a seguinte organização:

- No **Capítulo 2** é apresentado o estado da arte da área de recuperação por conteúdo, mostrando os principais métodos de acesso para dados concebidos em domínios métricos. É sumarizada a importância dos modelos de custo para estes métodos e apresentada uma visão de como é feita a extração de características para imagens e o uso de funções de distância. Como decorrência da extração de características, são mostrados alguns métodos existentes que tratam a redução da dimensionalidade destes dados. São apresentados também os tipos mais comuns de consultas por similaridade, finalizando com exemplos de aplicações que fazem uso delas.

- No **Capítulo 3** são apresentadas a definição formal de espaço métrico e algumas propriedades interessantes a este trabalho, como as aplicações lipschitzianas, a imersão de espaços métricos e os efeitos da maldição da alta dimensionalidade.

- No **Capítulo 4** é feito um estudo sobre várias funções de distância utilizadas para calcular a dissimilaridade dos dados complexos, analisando-se a faixa de valores de seu contradomínio e sua aplicabilidade. Também é feita uma análise sobre a abrangência de algumas funções, assim como sua homogeneidade no espaço dos dados.

- O **Capítulo 5** detalha uma nova técnica de imersão de espaços métricos que é utilizada para melhorar o desempenho de estruturas de acesso métrico para dados comple-

xos. Esta imersão permite utilizar propriedades específicas do espaço de imersão, onde definiu-se o conceito de Hiperplano Métrico, utilizado para otimizar o particionamento do espaço métrico.

- O **Capítulo** 6 descreve uma nova técnica usando mapeamentos de espaços baseando-se em aplicações lipschitzianas, deformando o espaço das distâncias entre elementos dentro de um espaço de alta dimensionalidade. Esta nova técnica atenua os efeitos negativos causados pela alta dimensionalidade melhorando o desempenho das consultas.

- O **Capítulo** 7 apresenta uma conclusão geral, mostrando os conceitos apresentados nesta tese e formulando conclusões e trabalhos futuros.

- O Apêndice A descreve a estrutura de dados *Slim-tree* e seu funcionamento, estrutura bastante utilizada durante o desenvolvimento do trabalho.

- Finalmente são apresentadas as **Referências Bibliográficas** utilizadas neste trabalho.

Capítulo

2

Recuperação de dados complexos por conteúdo

2.1 Introdução

Dados complexos, tais como os dados multimídia, são usualmente comparados por similaridade, pois em geral não podem ser comparados por operadores relacionais como os dados em domínios com relação de ordem total, nem por operações espaciais como os dados multidimensionais. Esses dados podem ser naturalmente representados em um espaço métrico, desde que seja definida uma métrica. Dados multidimensionais e adimensionais também podem ser representados em um espaço métrico, desde que uma métrica seja definida também. Vários MAM (métodos de acesso métrico) vêm sendo desenvolvidos para realizar cada vez mais eficientemente as consultas por similaridade, e muitos deles diferem pela estrutura de dados que utilizam ou pelas políticas adotadas junto às operações de atualização e consultas.

O trabalho pioneiro no desenvolvimento de MAM foi o trabalho de Burkhard e Keller [Burkhard & Keller, 1973]. A estrutura básica dos MAM leva a particionar o espaço em regiões usando elementos representantes, aos quais os elementos em cada partição estão associados. Cada partição tem um raio de cobertura e somente elementos dentro deste raio são associados ao representante (equivalente ao conceito de "bola" em espaços métricos, definida posteriormente na Seção 3.2.5).

Enquanto os métodos de acesso multidimensionais preocupam-se basicamente em minimizar o número de acesso a disco, os MAMs têm o intuito de reduzir também o número de cálculos de distância necessários para a execução de uma consulta, pois os elementos indexados são complexos e cada operação de comparação entre eles tende a ser custosa.

Durante a construção de um MAM, são calculadas as distâncias entre vários elementos a serem inseridos na estrutura. Essas distâncias são basicamente a única informação que se pode utilizar para estruturar os elementos. A partir daí, forma-se uma hierarquia de elementos representantes para guiar os operadores de busca para obter respostas mais rapidamente, sem precisar que se avaliem todos elementos indexados.

As distâncias calculadas, seja durante a construção da árvore ou quando da inserção de um novo elemento, podem ser armazenadas na estrutura. Assim, com as distâncias pré-calculadas pode-se estimar outras através da propriedade de desigualdade triangular. Quando cálculos de distância em domínios de dados complexos são computacionalmente custosos, o uso dessa propriedade estima os limites para determinadas distâncias, e pode evitar que cálculos de distância desnecessários sejam realizados, agilizando os operadores de busca. Esse mecanismo de poda por desigualdade triangular é apresentado a seguir.

Considere um espaço métrico $\mathbb{M} = < \mathbb{S}, d >$, um conjunto de elementos $S \subseteq \mathbb{S}$, o elemento de consulta $s_q \in \mathbb{S}$, um raio de consulta ξ e um elemento representante s_p. Uma consulta por similaridade visita os nós da árvore, a partir da raiz, descendo em ramos conforme o raio de consulta intercepte as regiões do espaço métrico, podendo descer em mais de um ramo por nível, uma vez que há intersecções entre as regiões.

Considerando essa consulta como na Figura 2.1, temos uma bola de consulta $B(s_q, \xi)$ que intercepta regiões do espaço métrico, como a figura mostra, interceptando a bola $B'(s_p, \xi_p)$. Mas, para saber se a bola $B''(s_r, \xi_r)$ contém parte da resposta, basta saber se

$$d(s_q, s_r) \leq \xi + \xi_r$$

O cálculo de distância $d(s_q, s_r)$ pode ser suprimido através da desigualdade triangular como segue. Note-se que $d(s_p, s_r)$ já foi previamente calculada e armazenada na árvore durante a inserção.

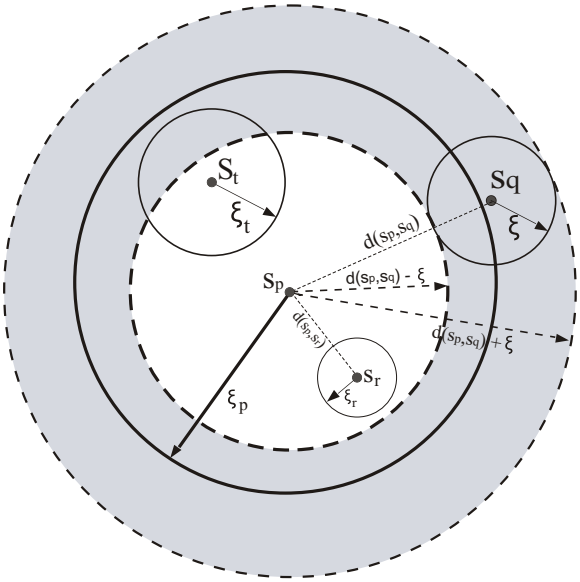


Figura 2.1: Utilização da desigualdade triangular para suprimir cálculos de distância.

Formalmente: Dado um domínio de elementos $\mathbb{S}$, uma métrica $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ e elementos $s_1, s_2, s_3 \in \mathbb{S}$, temos a desigualdade triangular:

$$d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2) \tag{2.1}$$

Esta equação permite inferir os limites superior e inferior de uma das distâncias conhecendo apenas os valores das outras duas distâncias.

Suponha-se que as distâncias entre s_1 e s_3 ($d(s_1, s_3)$) e entre s_2 e s_3 ($d(s_2, s_3)$) são conhecidas e deseja-se inferir o valor da distância entre s_1 e s_2 ($d(s_1, s_2)$), que será denotada por x para facilitar a compreensão. Usando a Equação 2.1 tem-se:

$$d(s_1, s_3) \leq x + d(s_3, s_2) \Rightarrow x \geq d(s_1, s_3) - d(s_3, s_2) \tag{2.2}$$

$$d(s_3, s_2) \leq d(s_1, s_3) + x \Rightarrow x \geq d(s_3, s_2) - d(s_1, s_3) \tag{2.3}$$

Sabendo que $x \geq 0$, as Equações 2.2 e 2.3 podem ser combinadas, formando:

$$d(s_1, s_2) \geq |d(s_1, s_3) - d(s_3, s_2)| \tag{2.4}$$

pois se $d(s_1, s_3) > d(s_3, s_2)$, o limite inferior será a equação 2.3, caso contrário, será a Equação

2.2.

Assim, baseando-se na Equação 2.4, pode-se descartar a região representada por s_r usando-se o limite superior da distância, como estimativa, e caso contrário, verifica-se a intersecção diretamente, calculando-se $d(s_q, s_r)$. Note-se que o mesmo conceito por ser aplicado nos nós folhas, sendo que a única diferença é que o raio de cobertura de s_r será zero. Esse conceito é o único mecanismo de poda atualmente empregado nos MAM utilizando propriedades do espaço métrico.

Métodos de acesso

É comum o uso de estruturas de dados para localização/recuperação de dados, e desde o princípio da era da informática buscam-se meios eficientes para realizar esta tarefa, tendo sido desenvolvidas estruturas muito boas. Uma primeira abordagem para resolver o problema de busca em um conjunto de dados, chamado "busca seqüencial" (*sequencial scan*), envolve a criação de um arquivo onde os dados estão dispostos seqüencialmente, o que é suficiente quando se manipulam pequenas quantidades de informações. Mas, quando lida-se com grandes quantidades de informações, essa falta de organização leva a consultas muito demoradas. Supondo que o item a ser procurado não exista, então o algoritmo deve percorrer todo o arquivo para garantir esta conclusão.

Assim sendo, é necessária a criação de estruturas especialmente desenvolvidas para localizar mais rapidamente os elementos armazenados. Como já mencionado, inicialmente houve a criação de estruturas para indexar elementos simples, como números e caracteres, ou seja, domínios de dados que possuem a relação de ordem total entre os elementos. Estruturas para esses domínios possuem um ótimo desempenho e admitem consultas com complexidade sublinear, sendo por isso muito utilizadas. O problema deste tipo de indexação para dados complexos é que nem sempre o elemento a ser indexado é um simples número, mas, por exemplo, uma coleção destes. Coleções de números não apresentam a propriedade de relação de ordem total.

Para domínios em que os elementos são coleções de números, em que toda coleção tiver o mesmo número de números (dimensionalidade), criam-se as estruturas de indexação mul-

tidimensionais. Para melhorar o desempenho desenvolveram-se estruturas para indexar dados espaciais, onde se utilizam estruturas geométricas n-dimensionais (hipercircunferências, hiper-retângulos, hipercubos etc.) para indexar os elementos [Gaede & Günther, 1998].

Métodos de acesso espaciais (MAE) podem ser usados para realizar buscas por similaridade, pois muitas das características extraídas dos elementos do domínio de dados, por exemplo de imagens, podem ser indexadas por tais métodos [Thomasian & Zhang, 2008] [Gao et al., 2009a] [Gao et al., 2009b] [Adler & Heeringa, 2008]. O ponto é que tais características atingem facilmente uma dimensão alta, a ponto de degradar o desempenho destes métodos em buscas por similaridade. Tal deficiência é conhecida como a "maldição da dimensionalidade" (*dimensionality curse*), um desafio para recuperar com eficiência elementos similares [Ramakrishnan et al., 2005]. Assim, métodos de acesso especializados foram propostos para lidar com altas dimensões de dados, como as árvores *SS-tree* [White & Jain, 1996], *X-tree* [Berchtold et al., 1996], *A-tree* [Sakurai et al., 2002], *EHD-tree* [Zhuang et al., 2008] e a $KRA^{+} - Blocks$ [Daoudi et al., 2009].

Estruturas de indexação concebidas para os espaços métricos, onde se utilizam uma métrica que mede a dissimilaridade entre os elementos do domínio são mais naturais para domínios de dados complexos. Tais estruturas utilizam as propriedades que podem reduzir o número de cálculos de distância em operações na estrutura, uma vez que estes cálculos são altamente custosos computacionalmente. Todavia, estas estruturas ainda estão limitadas apenas a estas propriedades, sendo necessário usar outros meios para conquistar maior desempenho.

A seção seguinte detalha e exemplifica os principais métodos de acesso métrico encontrados na literatura, explorando principalmente a estrutura de dados que cada método utiliza e as contribuições de cada um. Em seguida, a seção 2.4 resume as formas mais comuns para a extração de características de imagens, sendo estas utilizadas por métricas para avaliar a similaridade entre imagens.

2.2 Métodos de acesso métrico

Métodos de acesso métricos visam particionar o espaço de dados em várias regiões, escolhendo elementos representantes e agrupando os elementos restantes em sua volta. Eles podem ser classificados em duas categorias principais [Chávez et al., 2001]: aqueles baseados em métricas discretas e aqueles que lidam com métricas contínuas. MAM também podem ser classificados como estáticos ou dinâmicos, de acordo com seu suporte a inserções/remoções, após a criação da estrutura, como também podem ser classificados como armazenados em memória principal e armazenados em disco.

Muitos trabalhos já foram propostos como técnicas para responder às consultas por similaridade. O primeiro trabalho encontrado na literatura, que envolve indexação no espaço métrico, foi apresentado por Burkhard [Burkhard & Keller, 1973], onde são propostas três técnicas que particionam o espaço métrico hierarquicamente, gerando um MAM. Neste trabalho, as estruturas se aplicam quando as distâncias entre os elementos do domínio são discretas (números inteiros).

A primeira técnica constrói uma estrutura hierárquica da seguinte maneira: Escolhe-se s_r aleatoriamente, divide-se $S - \{s_r\}$ em $m+1$ subconjuntos $(S^0, S^1, S^2, ..., S^m)$, sendo que todo $s_j \in S^k$ se e somente se $d(s_j, s_r) = k$, onde $k = 0, 1, 2, ..., m$. A hierarquia é evidente quando a mesma técnica é aplicada recursivamente em cada subconjunto, criando uma árvore. Esta técnica é eficaz somente quando a distribuição de elementos é uniforme nos intervalos de distâncias, o que é difícil de impor em domínios de dados reais.

A segunda técnica é uma evolução da primeira, só que agora cada subconjunto tem associado um elemento representante com um raio de cobertura, sendo este a maior distância entre o representante e os restantes do mesmo subgrupo. Formalmente, a construção se dá da seguinte maneira: Divide-se S em s subconjuntos $(S^1, S^2, ..., S^s)$ tais que, para cada subconjunto S^i, existe um elemento representante sr_i onde para todo $x \in S^i, d(x, sr_i) \leq r_i$, onde r_i é o raio de cobertura daquela região. A estrutura hierárquica se forma a medida que a mesma técnica é aplicada aos elementos representantes obtidos em cada nível. Em seu trabalho, Burkhard e Keller não detalham como dividir o conjunto de elementos nem como eleger os elementos representantes de cada subconjunto.

A terceira técnica é similar à anterior, com o requisito adicional de que o diâmetro (distância máxima entre quaisquer dois elementos em um mesmo conjunto) de qualquer subgrupo seja menor que uma dada constante C, cujo valor é diferente para cada nível da estrutura. O conjunto satisfazendo esse critério é chamado de clique. A escolha do valor de C tem que garantir que todos os elementos do espaço estejam em pelo menos um dos cliques. Assim, forma-se a estrutura escolhendo o conjunto de cliques máximos, ou seja, cliques cujo diâmetro seja o maior possível, porém menor que C. Em seguida, um elemento de cada clique é escolhido aleatoriamente como representante daquele clique, para guiar ou podar pesquisas.

A *GH-tree* (*Generalized Hyper-plane tree*) [Uhlmann, 1991], a *VP-tree* (*Vantage-Point tree*)[Yianilos, 1993], a *MVP-tree* (*Multi-Vantage-Point tree*)[Bozkaya & Ozsoyoglu, 1997][Bozkaya & Ozsoyoglu, 1999], a *FQ-tree* (*Fixed Queries tree*) [Baeza-Yates et al., 1994] e a *GNAT* (*Geometric Near-Neighbor Access Tree*) [Brin, 1995] são desenvolvimentos que partem das técnicas em [Burkhard & Keller, 1973]. Todos esses MAM são denominados estáticos, pois exigem que todo o conjunto de dados esteja disponível no momento da construção das estruturas, não permitindo atualizações posteriores.

A primeira abordagem a armazenar distâncias pré-computadas, a fim de que a desigualdade triangular possa ser utilizada como mecanismo de poda, foi o trabalho de Shasha e Wang [Shasha & Wang, 1990]. O propósito aqui é evitar realizar os cálculos de distância, pois eles são computacionalmente custosos para elementos complexos. A técnica utilizada recai no uso de uma matriz quadrada de ordem n (onde n é o número de elementos indexados) que armazena as distâncias pré-computadas na construção da árvore. Esta abordagem é boa para domínios de baixa cardinalidade, mas quando o número de elementos é grande, fica inviável manter uma matriz de tal tamanho.

A *M-tree* [Ciaccia et al., 1997] foi o primeiro MAM dinâmico apresentado na literatura, sendo baseado na segunda técnica de [Burkhard & Keller, 1973]. Ela é uma estrutura balanceada pela altura que armazena os nós em registros de disco de tamanho fixo, possuindo dois tipos de nós: folha e índice. A *Slim-tree* [Traina et al., 2000] [C. Traina et al., 2002] (Veja Apêndice A) aperfeiçoou a *M-tree*, estabelecendo a primeira técnica de medição e redução da

sobreposição de subárvores capaz de operar em espaços métricos. Esta estrutura também é um MAM dinâmico, e possui ainda o recurso do algoritmo Slim-Down, que permite reorganizar a estrutura em qualquer etapa de sua existência para minimizar a sobreposição entre as subárvores em cada nó. A *M-tree* e a *Slim-tree* não dispõem da operação de remoção de elementos, o que é feito apenas marcando-se os elementos removidos, sem efetivamente removê-los. O trabalho de Yang [Yang et al., 2010] utiliza o algoritmo de *Slim-down* da *Slim-tree* na *SS-tree*, melhorando o desempenho da estrutura.

A *MM-tree* [Pola et al., 2007] foi criada para indexar dados complexos em memória para agilizar consultas por similaridade. A estrutura divide o espaço baseando-se em "bolas" que se intersectam no espaço métrico e possui técnicas que controlam o balanceamento da estrutura, além de algoritmos específicos para consultas que utilizam a proximidade das regiões como ordem de visita dos nós em cada nível da árvore. Deste modo, tem a vantagem de dividir o espaço métrico de modo efetivo e seu desempenho supera as várias estruturas comparadas já existentes na literatura. A *Onion-tree* [Carélo et al., 2009], uma extensão da estrutura *MM-tree* foi construída, com novas políticas de construção e particionamento do espaço. Ela divide o espaço em regiões baseando-se na multiplicidade do raio dos pivôs iniciais da *MM-tree*, produzindo assim mais regiões de abrangência do que a *MM-tree*, levando a um maior particionamento do espaço métrico e portanto melhor desempenho em consultas por similaridade.

Em [Santos et al., 2001] foi proposta uma técnica que permite criar novos MAM utilizando múltiplos representantes chamados "Omni-focos" para atuar como geradores de coordenadas para todos os elementos de um conjunto. Essas coordenadas podem ser indexadas utilizando-se qualquer ISAM, MAE, ou até mesmo busca sequencial, gerando toda uma nova família de MAM denominada Omni-family. Em [Traina et al., 2002b] foi proposto o uso de representantes universais junto a uma estrutura derivada da *Slim-tree* denominada *DF-tree*. Nessa estrutura, os representantes universais são usados junto com os representantes dos nós para aumentar a quantidade de podas em consultas por similaridade. Dois bons tutoriais discutindo os MAM existentes e sua aplicabilidade podem ser encontrados em [Chávez et al., 2001] e [Hjaltason & Samet, 2003].

Todos esses MAM são balanceados em altura, e isso limita a flexibilidade do método

para minimizar a sobreposição entre seus nós. A *DBM-tree* [Vieira et al., 2004] é o primeiro MAM dinâmico onde é possível diminuir a sobreposição existente em regiões do espaço de alta densidade de elementos através da flexibilização do balanceamento em altura da estrutura. Nessa estrutura é feito um compromisso entre a navegação em profundidade (minimizada pelo balanceamento em altura) e a navegação em largura (minimizada pela redução da sobreposição entre subárvores) para executar consultas por similaridade. Além disso, é proposto um algoritmo de reorganização dos nós chamado *Shrink*, derivado do *Slim-Down* proposto em [C. Traina et al., 2002] para a *Slim-tree*, porém executado em todos os nós da estrutura, e não apenas nos nós-folha, como é feito na *Slim-tree*. A $DBM^* - tree$ [Ocsa & Cuadros-Vargas, 2007] é uma extensão da *DBM-tree*, que mantém em cada nó uma matriz de distâncias que é usada nas operações de divisão dos nós na inserção e na consulta na estrutura utilizando-se exaustivamente da desigualdade triangular.

A tarefa de construir as estruturas a partir de um conjunto de dados é chamada *bulk-loading*. Estes métodos tem a vantagem de conhecer a distribuição das distâncias de maneira antecipada, e produzem melhores estruturas de indexação. O trabalho de Vespa [Vespa et al., 2010] apresenta novas abordagens para se realizar esta tarefa aplicada na *Slim-tree*. O trabalho de Aronovich [Aronovich & Spiegler, 2010] propõe uma técnica para lidar com inserções de porções de elementos na estrutura já criada, ao invés de inserções individuais.

A tarefa de remoção de elementos em um MAM envolve a re-estruturação de subárvores e é uma tarefa computacionalmente custosa. Ao invés de marcar o elemento como removido na estrutura, um trabalho que remove efetivamente elementos foi proposto por Bueno [Bueno et al., 2008].

Alguns trabalhos focam em técnicas de agrupamento ou paralelização para melhorar o desempenho das estruturas. A *CM-tree* [Aronovich & Spiegler, 2007], nome provindo de *clustered metric tree*, utiliza técnicas de agrupamento e *bulk-loading* para sua construção, além de manter as distâncias entre todos os elementos dentro de cada nó da estrutura. O trabalho de Lokoc [Lokoc, 2009] propõe uma nova técnica de construção da *M-tree*, baseado em paralelismo, destinado a sistemas em computadores com vários processadores (*multi-core processors*).

Algumas medidas de similaridade não são métricas, ou seja, não satisfazem as propriedades necessárias para formar um espaço métrico. Alguns trabalhos foram realizados com o intuito de possibilitar os métodos de acesso métrico a utilizarem estas medidas, utilizando funções de mapeamento [Liu & Hua, 2009] ou algoritmos para aproximação de consultas [Skopal & Lokoč, 2008].

Um trabalho sobre a utilização de abstração de generalização em espaços métricos foi o trabalho de [Pola et al., 2006]. Nele, foi desenvolvida uma técnica que permite que várias métricas sejam combinadas em uma única estrutura de indexação, onde os elementos indexados podem conter diferentes tipos de características associadas. Esta técnica envolve indexar elementos em um *Domínio Métrico Generalizável*, onde elementos são agrupados numa hierarquia em que cada nível pode ter associadas diferentes métricas e diferentes características para cada elemento.

2.3 Modelos de custo

A análise teórica dos diversos métodos de acesso é muito importante para o desenvolvimento de um SGBD, pois ela fornece os meios para entender e ajustar os métodos para diferentes tamanhos e tipos de conjuntos de dados, além dos meios para comparação dos diversos métodos. As operações de consultas podem ser substancialmente melhoradas utilizando técnicas de otimização. Porém, para tal, é essencial a existência de técnicas para realizar uma estimativa dos métodos de acesso [Böhm, 2000]. Os primeiros trabalhos sobre modelos de custo para dados multi-dimensionais estão em [Kamel & Faloutsos, 1993] [Faloutsos & Kamel, 1994] [Theodoridis et al., 2000] [Papadopoulos & Manolopoulos, 1997] [Böhm, 2000].

Em [Kamel & Faloutsos, 1993] é apresentado um modelo de custo de acesso a disco para a consulta Rq em função das características geométricas dos nós de *R-trees* [Guttman, 1984] indexando dados uniformemente distribuídos no espaço. Uma extensão é apresentada em [Faloutsos & Kamel, 1994], na qual se assume que os nós são em forma de hiper-cubos n-dimensionais. O correspondente modelo de custo é baseado no conceito de dimensão fractal (ou dimensão intrínseca) do conjunto de dados. Este modelo é aplicável a dados que não

estão uniformemente distribuídos no espaço, mas que possuem correlação entre os dados, o que é a situação usual em dados reais. Este modelo utiliza a dimensão fractal do conjunto de dados, que é um número que expressa a correlação entre os dados.

Em [Belussi & Faloutsos, 1995] é apresentado um modelo de previsão de seletividade em consultas Rq e junções espaciais para consultas polarizadas (*biased* - a distribuição das consultas no espaço de dados segue a distribuição dos elementos de dado no espaço) em SAM utilizando a Teoria dos Fractais. Em [Theodoridis et al., 2000] é apresentado um modelo de custo de acessos a disco para a consulta Rq sem conhecimento prévio das características na *R-tree*. Este modelo explora informações estatísticas da distribuição dos dados, apresentado em forma de histogramas de densidade. Ele é o único modelo de custo dependente da consulta. Os demais fornecem estimativas de custo médio para todo o conjunto.

O desempenho de consultas $kNNq$ para o caso específico de $k = 1$, é considerado em [Papadopoulos & Manolopoulos, 1997] [Böhm, 2000]. O trabalho de Papadopoulos [Papadopoulos & Manolopoulos, 1997] é uma extensão dos trabalhos apresentados em [Faloutsos & Kamel, 1994][Belussi & Faloutsos, 1995], onde a análise de custo de acessos a disco é aplicada em espaços Euclidianos bi-dimensionais para *R-trees*, baseado na dimensão fractal do conjunto de dados e considerando que o elemento central da consulta pertence ao conjunto de dados indexado. Em [Böhm, 2000] é introduzido um modelo de custo de acessos a disco e de número de cálculos de distância para dados uniformemente distribuídos em espaços Euclidianos de alta dimensão para a *R-tree*, não levando em consideração a taxa de sobreposição entre os nós da estrutura.

É importante notar que todos os modelos de custo citados anteriormente foram desenvolvidos apenas para MAE, e que nenhum deles podem ser aplicados a domínios métricos.

Os primeiros modelos de custo para MAM são os propostos em [Ciaccia et al., 1998] [Traina Jr et al., 1999] [Ciaccia et al., 1999] [Jr. et al., 2000a] [Traina et al., 2000]. O primeiro trabalho envolvendo modelos de custo para estruturas que operam em espaços métricos foi o de [Ciaccia et al., 1998], que é uma adaptação do modelo de [Berchtold et al., 1997] para espaços métricos. Nesse trabalho, a distribuição das distâncias entre os elementos e estatísticas obtidas da própria estrutura são utilizadas para a previsão de custos. Esse modelo permite otimizar o tamanho de registros de disco para minimizar custos nas consultas, além

de estimativas do número de cálculos de distância e de acessos a disco para as consultas Rq e $kNNq$. Este modelo fornece estimativas de custo médio para as consultas e é eficaz apenas para dados em espaços homogêneos. O modelo de [Ciaccia et al., 1999] é uma extensão de [Ciaccia et al., 1998], propondo um modelo de custo que depende da consulta.

Em [Traina Jr et al., 1999] [Jr. et al., 2000b] [Traina et al., 2000] a dimensão fractal do conjunto de dados e informações estatísticas da estrutura são usadas para propor um modelo de estimativa de custo para o número médio de acessos a disco. Este modelo é o único que leva em consideração a taxa de ocupação dos nós da estrutura. No trabalho de Baioco [Baioco et al., 2007] é apresentado um modelo de custo para estimar o número de acessos a disco e o número de cálculos de distância para processar consultas por similaridade em MAM.

2.4 Extratores de características

Um dos domínios de dados complexos que mais tem atraído a atenção dos pesquisadores em bases de dados, dada a grande necessidade de apoio que ele requisita dos SGBD, é o domínio de imagens.

Existem diferentes formas de analisar a similaridade entre duas imagens. Dependendo da categoria de recuperação (nível de abstração) desejada, a extração de características pode tomar um rumo bem distinto. Tais categorias referem-se a quais atributos são de relevância, e podem se referir a: atributos visuais, como cor, forma, textura ou mesmo a uma combinação destes; atributos lógicos, como a identificação de elementos (p. ex. selecione as imagens que contém carros); atributos semânticos, como a identificação de emoções humanas (selecione imagens que expressem alegria). Nesta seção serão citadas algumas técnicas de extração para atributos visuais de imagens.

Características visuais (cor, forma ou textura) são naturalmente utilizadas para extração de características. Várias técnicas foram propostas para extrair características sobre cada característica visual, sendo que muitas delas são especializadas para determinados domínios de imagens, nos quais a especificidade das operações de comparação aumenta.

Análise da cor

São encontrados na literatura diversos métodos de recuperação de imagens baseados na similaridade de cor, sendo que quase todos eles compartilham a mesma idéia: para cada imagem contida na base de dados é calculado seu histograma de cores.

A identificação de uma imagem através da característica "cor" é realizada geralmente pela construção de um histograma em que são calculados o número de pixels da imagem com cada cor. Um tipo de medida é a análise dos tons de cinza, muito utilizada para imagens médicas, onde a imagem é analisada e seu histograma (conhecido como *gray level histogram* ou como *brightness histogram*) é gerado e utilizado para comparação nas buscas. Para comparar computacionalmente duas imagens, utilizam-se os histogramas das imagens e a comparação usualmente é realizada mediante alguma norma L_p, em geral L_1.

Em uma consulta, o usuário tanto pode especificar a proporção desejada de cada cor (em porcentagem, por exemplo), quanto submeter uma imagem exemplo para a qual também será calculado o histograma. Para ambos os casos recuperam-se as imagens da base cujo histograma mais se aproxime daquele dado (ou calculado) para a pesquisa. A técnica de comparação de histogramas mais utilizada foi proposta por Swain e Ballard [Swain & Ballard, 1991]. Alguns métodos propostos apresentam melhorias para esta técnica. Dentre eles encontram-se o uso de histograma de cores acumulativo [Stricker & Orengo, 1995], o uso de análise por cor baseada em região [Carson et al., 1997] e o uso de histogramas métricos [Traina et al., 2002a, Traina et al., 2003a]. Trabalhos recentes buscam aprimorar a qualidade da recuperação baseada em cor [Fanjun et al., 2009] [Junjun et al., 2008].

A comparação baseada em cor é computacionalmente simples, linear e pouco sensível a pequenas alterações na imagem (movimentações). No entanto, apresenta alguns pontos a serem considerados. O fato é que duas imagens bem distintas podem possuir histogramas de cores semelhantes, uma vez que o método é estatístico. Outro ponto é que freqüentemente o número de cores é elevado (≥ 256), gerando vetores de características de dimensão alta, o que se torna um grave problema para as estruturas de indexação ("maldição da alta dimensionalidade" [Volnyansky & Pestov, 2009]).

Devido à pouca capacidade de discriminar imagens distintas que possuem uma densidade de cor semelhante, os histogramas devem ser utilizados em conjunto com outras técnicas de extração de características. Por outro lado, como o uso de histogramas é simples e relativamente pouco custoso, é comum que se empreguem histogramas como um primeiro passo de filtragem, que pode eliminar muitas imagens de um processamento posterior, reduzindo a quantidade daquelas que devem ser submetidas a operação de comparação mais elaborada e custosa. É importante destacar que qualquer imagem pode ter seu histograma calculado por uma operação com custo linear no tamanho da imagem. Além disso, é possível reduzir a resolução de cor de uma imagem, de maneira que o histograma de cor é um meio de comparação que pode ser aplicada a qualquer par de imagens.

Análise da textura

A capacidade de lidar com texturas pode ser útil para distinguir regiões de cores similares (por exemplo, o mar e o céu, ou folhagem e grama), identificando um padrão de variações dessas cores. Várias técnicas para extrair informações de textura podem ser encontradas na literatura. A mais conhecida analisa conjuntos de pares de pixels da imagem e monta estruturas com informações estatísticas, a partir das quais são extraídas as características como periodicidade, nível de contraste, rispidez, direcionalidade, regularidade, entre outras. Como exemplo, duas estruturas tradicionalmente utilizadas em análise por textura são as "Matrizes de co-ocorrência" (*Coocurrence Matrices*) e as "Matrizes *Run-Lengths*". Alguns métodos de análise alternativos incluem o uso de "filtros de Gabor" [Manjunath & Ma, 1996] e fractais [Kaplan et al., 1998]. Pesquisas de imagem por textura são formuladas de maneira similar àquelas baseadas em cor, selecionando-se exemplos de texturas a serem procuradas ou fornecendo uma imagem exemplo para consulta.

Uma matriz de co-ocorrência pode ser descrita como segue. Dada uma imagem I com um conjunto discreto de tons de cinza C, define-se a matriz de co-ocorrência $P_{d,\phi}(i,j)$, onde cada elemento (i,j) corresponde a um número inteiro que indica quantas vezes um pixel p_1 de nível de cinza i aparece distante de um pixel p_2 de intensidade j por uma distância d e um ângulo ϕ (verificando-se os dois sentidos, horário e anti-horário). Formalmente, cada elemento (i,j) de $P_{d,\phi}(i,j)$ indica quantas vezes ocorre $p_1 = p_2 + (d\cos\phi, d\sin\phi)$, onde

$I(p_1) = i$ e $I(p_2) = j$. Logo, as matrizes de co-ocorrência são matrizes quadradas e simétricas em relação à diagonal principal, ou seja, $P_{d,\phi}(i,j) = P_{d,\phi}(j,i)$.

Diversas medidas podem ser extraídas de uma matriz de co-ocorrência. Haralick [Haralick et al., 1973] propõe várias delas. As medidas energia e entropia dão uma indicação do comportamento da textura em relação à sua uniformidade e periodicidade. A medida de contraste analisa os valores da matriz com ênfase nos elementos mais distantes da diagonal, ou seja, os pontos cujos níveis de cinza possuem maior distinção entre si, o que vale, de fato, como um indicador do nível de contraste da textura. A homogeneidade possui a mesma tendência, porém, com sentido inverso.

Um exemplo de utilização de textura para a recuperação de imagens pode ser encontrado em Felipe [Felipe et al., 2003] onde, baseando-se em matrizes de co-ocorrência, define-se um novo descritor chamado gradiente e indica uma combinação ótima de três descritores (gradiente, homogeneidade e entropia) para recuperar imagens por similaridade com alta precisão.

A grande desvantagem da utilização das matrizes de co-ocorrência na extração de características da textura está relacionada com seu alto custo computacional. Computar várias matrizes de grandes dimensões nem sempre é uma alternativa factível. Também, a escolha adequada de seus parâmetros d e ϕ depende muitas vezes de um conhecimento prévio acerca da qualidade das imagens ou de seu domínio, o que nem sempre é possível, ou desejável, em determinadas aplicações.

Trabalhos recentes focam determinar modelos de classificação efetivos para imagens bi-dimensionais e tridimensionais [Qi et al., 2008] [Jian et al., 2008].

Análise por forma

Muitos estudos afirmam que elementos reais são identificados pelo sistema de visão humana primordialmente pela forma [Biederman, 1987, Levi & Klein, 2000, Loffler et al., 2003], justificando a pesquisa em novas técnicas de extração de características baseadas neste atributo.

Nesta abordagem, uma série de características de forma são calculadas e armazenadas para cada elemento encontrado em uma imagem para cada imagem da base de dados. Uma propriedade desejável do conjunto de características utilizado é que ele seja invariante para

elementos de mesma natureza que estejam em posição, rotação e escala diferentes (invariância às transformações geométricas), e que possam descrever adequadamente a forma do elemento mesmo quando a imagem contém ruídos. Quanto à classificação dos métodos de extração de características de forma, a mais comum apresenta os métodos baseados em contornos, que analisam apenas os contornos dos elementos e os métodos baseados em região, que analisam o elemento como um todo, levando em consideração também os pontos de seu interior.

A representação do atributo "forma" dos elementos presentes nas imagens é dada por vetores de características que podem possuir dimensões distintas. Assim, se definida uma métrica entre formas, é necessário um MAM que pode ser usado para indexar imagens com estas características. Trabalhos que analizam forma para automatizar processos de segmentação ou de detecção de contorno são bastante estudados na literatura [Bai et al., 2008] [Falomir et al., 2009] [Xiao et al., 2008] [Fu et al., 2008b].

Gap semântico

Um problema bem conhecido na área de CBIR, e citado por muitos autores, diz respeito à incapacidade das características de baixo nível em descrever o conhecimento semântico embutido nas imagens. A representação matemática da imagem na qual os sistemas CBIR são baseados está bem mais relacionada à estrutura matricial da imagem do que à representação perceptual que uma pessoa faz daquela imagem, incluindo o significado dos objetos e a complexa rede de relacionamentos que pode existir entre eles. A esta discrepância de representações da imagem é dado o nome de *gap* semântico, que, embora seja abordado em diversos trabalhos, ainda continua sendo um problema em aberto e um dos maiores desafios em CBIR. No trabalho de Liu et al. [Liu et al., 2007], por exemplo, são propostas cinco abordagens para suprimir o problema do *gap* semântico nas características de baixo nível, incluindo a utilização de técnicas de aprendizado de máquinas e segmentação de imagens, a utilização de realimentação de relevância por parte do usuário, e a definição de padrões semânticos para classificação das imagens. É também coerente pensar que o aumento da especialização de um sistema CBIR com relação ao domínio de imagens considerado causa o estreitamento do *gap* semântico, porque possibilita o aproveitamento de conhecimento prévio do domínio [Müller et al., 2004].

2.5 Redução de dimensionalidade

A alta dimensionalidade, comum em conjuntos de objetos reais, aumenta a complexidade de tarefas de manipulação de dados. Em consequência, a eficiência de técnicas desenvolvidas para executar estas tarefas é afetada significativamente o que é chamado de "maldição da dimensionalidade". O aumento do número de atributos degrada a performance dos algoritmos não só em tempo de processamento, mas também na qualidade dos resultados. Na área de mineração de dados, a "maldição da dimensionalidade" é um problema enfrentado tanto em atividades preditivas quanto descritivas. Por exemplo, a grande quantidade de atributos aumenta o espaço de busca para a definição de modelos de classificação, reduzindo a precisão na discriminação de objetos em classes distintas [Aggarwal, 2005]. Além disso, os objetos em um espaço de alta dimensionalidade tendem a ficar mais esparsos e as distâncias entre pares de objetos tendem a ser muito próximas [Beyer et al., 1999b], o que dificulta a detecção eficiente de agrupamentos [Aggarwal & Yu, 2002].

Para reduzir os efeitos da "maldição da dimensionalidade", técnicas de redução de dimensionalidade podem ser aplicadas antes do processo de mineração de dados, tranformando os dados para a análise. Estas técnicas têm como objetivo representar um conjunto de dados com alta dimensionalidade em um espaço de dimenssão inferior à original n, mantendo as propriedades intrínsecas do conjunto tanto quanto possível e reduzindo a complexidade das tarefas realizadas na etapa de mineração.

Em geral, a redução de dimensionalidade é factível, pois conjuntos de dados reais são usualmente caracterizados pela distribuição não uniforme e pela existência de muitas correlações entre os atributos [Faloutsos & Kamel, 1994]. Se dois ou mais atributos no conjunto estão correlacionados, existe um mapeamento que permite que o valor de um deles seja determinado ou muito aproximado pelo outro, ou seja, existe um número reduzido de valores que o segundo atributo pode assumir em função do primeiro. Logo, uma das abordagens para redução de dimensionalidade é eliminar atributos envolvidos em correlações. Note-se que, quando os atributos de um conjunto estão correlacionados, os dados tendem a ocupar um subespaço de dimensão menor que n. Assim, encontrar uma representação para este subespaço é o objetivo da redução de dimensionalidade.

Existe uma variedade de técnicas propostas para a redução da dimensionalidade baseando-se em diferentes princípios. Neste documento, serão mostrados apenas os conceitos básicos envolvidos nas diferentes abordagens. A Figura 2.2 mostra as diferentes abordagens que podem ser seguidas no processo de redução.

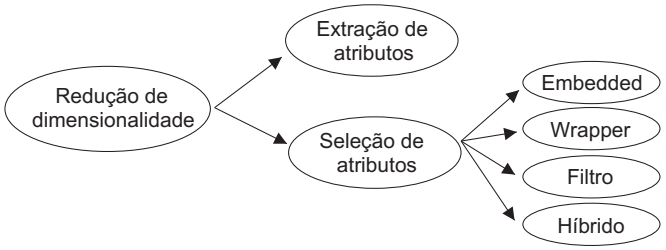


Figura 2.2: Diferentes abordagens para a redução de dimensionalidade de conjuntos.

A **extração de atributos** tem por objetivo definir um subespaço de dimensão menor ou igual à do espaço de atributos original, mapeando o conjunto de dados neste subespaço e procurando manter as características fundamentais do conjunto. A extração de atributos é baseada na transformação do espaço de atributos original em um espaço de dimensão mais baixa. Métodos clássicos de extração de atributos são o PCA (*Principal Component Analysis*) e SVD (*Singular Value Decomposition*), descritos no trabalho de Haykin [Haykin, 1994]. Em geral, técnicas baseadas nestes métodos reduzem a dimensão dos dados originais por meio de transformações lineares, com o objetivo de identificar direções ortogonais de máxima variância. Os dados são mapeados no espaço de dimensão menor formado pelo subconjunto de componentes de variância mais alta. O método ICA (*Independent Component Analysis*), também baseado em transformações lineares, tenta minimizar a dependência entre os componentes de representação do conjunto de dados [Hyvärinen, 1999].

Os métodos PCA, SVD e ICA eliminam apenas correlações lineares e, portanto, são pouco eficientes na redução de dimensionalidade de conjuntos caracterizados pela existência de correlações não-lineares. As propostas para extração não-linear de atributos encontradas na literatura seguem princípios diversos. Algumas delas são generalizações não-lineares de PCA que utilizam, por exemplo, redes neurais autoassociativas (*auto-associative feed-forward neural networks*) [DeMers & Cottrell, 1993] [Kramer, 1991], funções kernel (kernel PCA) [Schölkopf et al., 1998] e curvas principais (*principal curves*) [Chang & Ghosh, 1998]. Numa abordagem diferente, os trabalhos apresentados por Roweis [Roweis & Saul, 2000] e

Tenenbaum [Tenenbaum et al., 2000] são baseados em mapeamento de atributos com preservação de distâncias locais.

A **seleção de atributos** (*feature selection*) tem como objetivo encontrar, no conjunto de dados original, um subconjunto de atributos relevantes, mantendo as propriedades dos dados e minimizando a perda de informação. A seleção, ao contrário da extração de atributos, define um espaço de dimensão mais baixa formado por atributos do espaço original, o que facilita a interpretação do resultado alcançado. A seleção de atributos é útil para encontrar um subconjunto de atributos interessantes em um conjunto de dados que contenham atributos irrelevantes ou redundantes. Entretanto, encontrar este subconjunto é, geralmente, um problema complexo computacionalmente, caro e muitas vezes intratável, pois um conjunto de dados com n atributos requer a análise de 2^n possibilidades [Liu & Yu, 2005].

As técnicas de seleção de atributos são tipicamente divididas em três modelos principais, de acordo com o tipo de interação realizado com algoritmos de aprendizado de máquina, definidas no trabalho de Blum [Blum & Langley, 1997]: *wrapper*, *embedded* e filtro. Em trabalhos mais recentes, uma quarta abordagem, chamada de híbrida, combina os modelos *wrapper* e filtro, explorando seus diferentes mecanismos de avaliação em fases distintas do processo de busca (Liu e Yu, 2005). O modelo *embbeded* é usualmente aplicado em atividades de seleção de atributos supervisionada. As técnicas *wrapper*, filtro e híbrida, por outro lado, vêm sendo aplicadas tanto para seleção de atributos supervisionada como para não-supervisionada.

Em uma abordagem diferente, mais voltada para a área de bases de dados, Traina [Jr. et al., 2000c] propõem o algoritmo FDR (Fractal Dimensionality Reduction), que determina a relevância dos atributos por meio do impacto que cada um deles causa na dimensão fractal do conjunto de dados. O algoritmo utiliza a busca por *sequential backward elimination*, em que são descartados os atributos cuja remoção causa menor alteração na dimensão fractal. O resultado final é uma lista de atributos em ordem crescente de relevância. O FDR é um algoritmo de seleção de atributos de propósito geral e, como tal, pode ser aplicado a conjuntos de dados com ou sem informação de classe e em etapas de pré-processamento em qualquer atividade de manipulação de dados. A identificação de grupos de atributos correlacionados, estendendo a abordagem inicial de Traina, foi proposta por

Sousa [Sousa et al., 2007].

2.6 Funções de distância

Estruturas de indexação concebidas para o espaço métrico se utilizam de uma função de distância que mede a dissimilaridade entre os elementos do domínio. Como visto, nestes domínios, a similaridade entre os elementos é o fator mais importante. Portanto, os métodos de acesso métrico são construídos diretamente sobre medidas de similaridade entre os objetos e usam fundamentações matemáticas para realizar podas com base nas distâncias. Métodos de acesso que utilizam esta abordagem são chamados baseados em distâncias (*distance-based structures*), onde algumas distâncias podem ser armazenadas na estrutura de dados que organiza os dados para agilizar as buscas através da utilização da desigualdade triangular (em geral o limite inferior desta) para auxiliar a "poda" de subárvores.

A Figura 2.3 mostra a utilização da função de distância pelos métodos de acesso métrico. A função compara cada par de elementos do conjunto, e retorna o nível de similaridade entre eles. Quanto mais próximo de zero, mais semelhantes os elementos são, e vice versa. O Capítulo 4 exemplifica diversas funções de distâncias encontradas na literatura, além de compará-las quanto a sua abrangência e correlação de limites de valores.

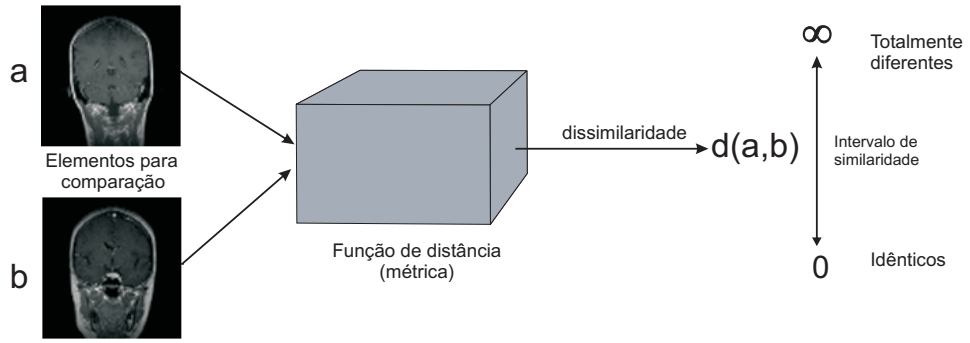


Figura 2.3: O uso de uma função de distância.

2.7 Consultas por similaridade

Consultas por similaridade devem ser feitas sobre elementos em um espaço métrico, pois requerem a existência de uma métrica. Os tipos de consultas por similaridade mais comuns são as consultas por abrangência (*range queries*) e as consultas pelos k vizinhos mais próximos (*k-nearest neighbor queries*). Além delas, existem algumas menos comuns, como consulta aos vizinhos mais próximos reversos, também descrita a seguir. Observe-se que a desigualdade triangular pode ser usada para evitar que cálculos de distância desnecessários sejam feitos nestas consultas, como indicado na Seção 2.1, o que pode ser alcançado pelos MAM sobre outras estruturas espaciais.

Definição 2.1 *Consulta por abrangência (range query - Rq): Uma consulta por abrangência recebe como parâmetro de entrada um elemento de referência s_q e um raio de cobertura ξ formando uma bola de consulta $B(s_q, \xi)$. A resposta deve incluir todos elementos s_j que interceptam a bola de consulta. Formalmente:*

$$Rq(s_q, r_q) = \{s_j \mid d(s_j, s_q) \leq \xi\}, \ s_q, s_j \in S. \tag{2.5}$$

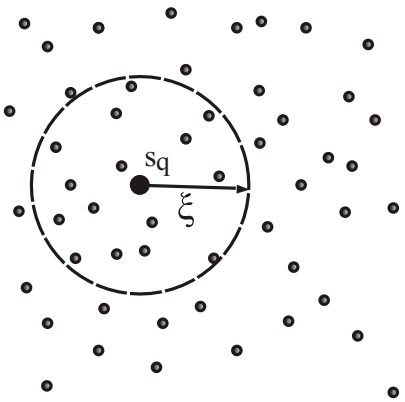


Figura 2.4: Exemplo de consulta por abrangência.

A Figura 2.4 exemplifica uma consulta deste tipo num espaço bidimensional com a métrica euclidiana L_2 (veja Seção 3.2.2). Os elementos contidos pelo raio de cobertura r_q compõem a resposta. Vale lembrar que não é necessário que o elemento de consulta pertença ao conjunto de dados de pesquisa, devendo apenas pertencer ao mesmo domínio dos dados.

Definição 2.2 *Consulta aos k vizinhos mais próximos (k nearest neighbor query - kNNq): Este tipo de consulta recebe um elemento de referência s_q e o número k de vizinhos mais próximos que se deseja recuperar. Formalmente (mais detalhes em [Ferreira et al., 2009]):*

$$kNNq(s_q, k) : S = \{s_i \in S \mid \forall\, s_j \in S - S', d(s_i, s_q) \leq d(s_j, s_q)\}, \tag{2.6}$$

onde $S' = \emptyset$, se $i = 1$ e $S' = \{s_1, \ldots, s_{i-1}\}$, se $1 < i \leq k$.

A Figura 2.5 exemplifica uma consulta deste tipo num espaço bidimensional com a métrica euclidiana L_2 novamente. Na figura, a consulta teve como entrada o elemento de consulta s_q e o valor de k igual a 4. Apenas os elementos ligados por traços (os mais próximos) compõem a resposta.

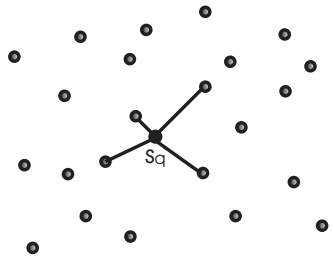


Figura 2.5: Exemplo de consulta por vizinhos mais próximos.

Um detalhe que deve ser tratado na implementação deste algoritmo é o tratamento de empates. Suponha que seja definida uma consulta pelos dois vizinhos mais próximos a partir de um elemento central de consulta, mas no entanto existem três elementos igualmente distantes do elemento central e eles são os mais próximos a ele. Nesse caso, deve ser dada uma opção de escolha a quem utiliza a estrutura de indexação se toda a lista de empates deve ser incluída no resultado (retornando $\geq k$ elementos) ou deve-se retornar exatamente o número pedido de elementos k escolhendo-se aleatoriamente quais dos elementos espatados retornam.

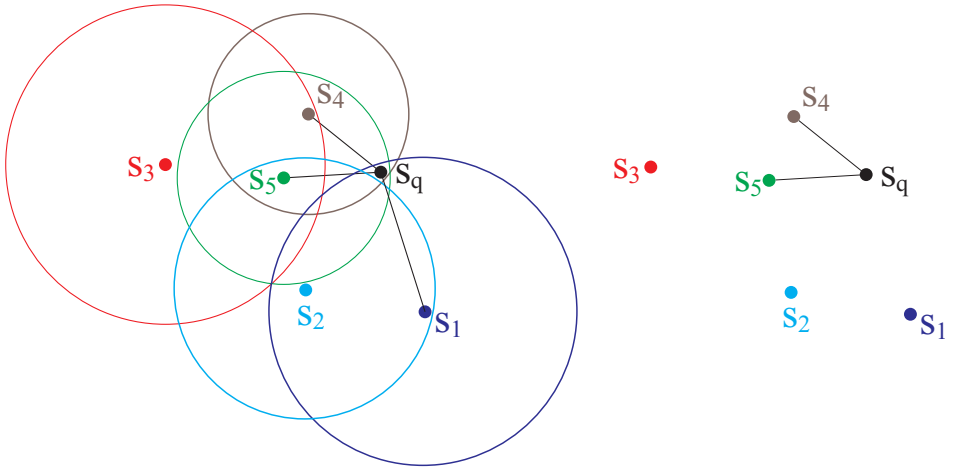
Definição 2.3 *Consulta aos k vizinhos mais próximos reversos(reverse k nearest neighbours query - RkNNq): Este tipo de consulta retorna elementos na região do elemento de consulta*

que o possuem na lista de seus k vizinhos mais próximos. Pode-se expressar esta consulta da seguinte maneira:

$$RkNNq(s_q, k) = A = \{s_i \in S, s_q \in kNN(s_i, k) \wedge \forall s_j \in (S - A) : s_q \notin kNN(s_j, k)\}$$

Primeiramente definido por Korn e Muthukrishman [Korn & Muthukrishnan, 2000], para o caso de k=1 (*reverse nearest neighbour*), esta consulta descreve a região de influência de um elemento de consulta sobre o conjunto de dados.

Comparando-se o $kNNq$ e o $RkNNq$, pode-se considerar que eles são assimétricos, pois nem todos elemento presente na lista de resultados de um $RkNNq$ está na lista de um $kNNq$. Considerando-se que o elemento de consulta s_q não esteja presente na base de dados, podemos comparar os dois operadores de consulta usando como exemplo a Figura 2.6. Seja o conjunto resposta representado pela ligação dos elementos a s_q, na figura. Nota-se que, para este exemplo, o conjunto resposta da consulta $RkNN(s_q, 2)$ em (a) é $\{s_1, s_5, s_4\}$ pois cada um desses elementos têm s_q como um de seus 2 elementos mais próximos, enquanto o conjunto resposta da consulta $kNN(s_q, 2)$ em (b) é $\{s_5, s_4\}$.



(a) Consulta RkNN com k = 2 (b) Consulta kNN com k = 2

Figura 2.6: Exemplo comparativo entre as consultas kNN e RkNN para um mesmo conjunto de elementos.

2.8 Aplicações

Aplicações que manipulam dados complexos (como dados multimídia) devem fornecer mecanismos para responder a consultas por similaridade. Definir a similaridade vem sendo um desafio na questão de avaliar se ela é apropriada para o domínio de dados em questão. A seguir são mostradas algumas aplicações que manipulam tipos de dados complexos e usam consultas por similaridade.

Tipos de dados como imagens, vídeo, áudio e textos são considerados dados multimídia e podem ser armazenados em **bases de dados multimídia** [Traina & Jr, 2003]. Em aplicações tradicionais, é comum a inclusão de descrições textuais ou palavras chaves associadas a estes tipos de dados, para posterior consulta, mas isso leva a erros oriundos da interpretação individual de cada usuário ao inserir dados, além do fato de que outras características podem deixar de ser descritas sem se perceber. Assim, um método automático é melhor para extrair características dos dados para posterior consulta por similaridade. Uma métrica específica deve ser definida para avaliar em primeiro lugar a similaridade a partir dessas características extraídas. Por exemplo, para o domínio de imagens, atributos tais como forma, textura e cor, entre outros, são extraídos das imagens para a composição de vetores de características e uma métrica deve ser definida para avaliar estas características [Datta et al., 2008] [Smeulders et al., 2000]. Estas características são extraídas e armazenadas na base de dados, onde são analizadas posteriormente pelos métodos de acesso para comparação em consultas por similaridade.

Assim, consultas do tipo "recupere as cinco imagens mais parecidas com a imagem A" são realizadas de modo que o método de acesso verifique quais são as cinco imagens da base de dados que possuem o menor valor de distância em relação à imagem de consulta (A) e as incluam no conjunto resposta. Se a seletividade da consulta for pequena, o que normalmente é, um índice feito utilizando um método de acesso métrico agiliza e muito este processo, não precisando verificar toda a base de dados seqüencialmente para responder tal consulta (*sequencial scan*), a menos que não haja um índice criado [Barioni, 2002].

Seqüências de DNA podem ser interpretadas como fragmentos de texto e formam **bases de dados genéticas**, uma vez que as quatro bases nitrogenadas que compõe as seqüências

de DNA podem ser representadas pelas letras *A, G, C e T* [Hunt et al., 2001]. Consultas submetidas por estas aplicações demandam buscas por ocorrências de uma seqüencia de letras em outra seqüencia maior, normalmente para a detecção de anomalias genéticas. Neste caso, a aplicação da métrica L_{Edit} se torna inviável, tornando necessário o uso de outros tipos de abordagens [Russo et al., 2008] [Chen et al., 2009].

Bases de séries temporais podem ser qualquer coleção de dados usualmente multi-dimensionais, feita seqüencialmente no tempo. Consultas buscam por padrões similares ao longo de um coleção de sequências. Alguns trabalhos usam diferentes formas de comparação de sequências, como por exemplo utilizando escalas uniformes ao invés da distância eucliana [Euachongprasit & Ratanamahatana, 2008]. O trabalho de Kim [Kim et al., 2006] propôs um modelo que recupera subsequências cuja "forma" é similar à sequência de consulta, utilizando árvores de sufixo para indexação. Alguns outros trabalhos focam o estudo de diferentes formas de comparação, propondo adaptações em funções de distância para melhorar a qualidade da recuperação, como funções do tipo *warping* (*time warping distances*) [Marteau, 2009] [Athitsos et al., 2008] [Keogh & Ratanamahatana, 2005] [Fu et al., 2008a] [Keogh et al., 2009].

Algumas outras aplicações que se pode citar são: **reconhecimento de fala** [Wechsler et al., 2000], onde buscam-se padrões vocais similares (freqüentemente através de transformadas de *Fourier*) de uma base de padrões vocais; **vídeos**, onde o desafio recai em recuperar sequências de vídeos similares [Zhou et al., 2007]; **detecção de cópias**, onde plágios podem ser detectados buscando sentenças similares em um grande repositório de documentos.

Um grande desafio é incorporar consultas por similaridade em SGBDs relacionais. A linguagem SQL não tem suporte direto aos operadores por similaridade, mas trabalhos foram realizados para suprir esta necessidade. Um sistema interpretador chamado SIREN (*Similarity Retrieval Engine*) inclui o suporte de consultas por similaridade na linguagem SQL. O sistema permite definir campos do tipo imagem (*STILLIMAGE*) em tabelas e permite que consultas por similaridade sejam feitas nesses atributos [Barioni et al., 2006]. Na mesma linha de pesquisa, o trabalho de Guliato [Guliato et al., 2009] propõe uma extensão para manipular imagens no SGBD PostgreSQL, chamada PostgreSQL-IE, estendendo as

funcionalidades da linguagem SQL com funções para manipular vetores de características de imagens, assim como consultas por similaridade.

2.9 Conclusão

Os SGBDs tradicionais suportam consultas onde a propriedade de relação de ordem total é atendida por todos os dados e portanto pode ser utilizada como mecanismo de busca e comparação nos métodos de acesso. Porém, em aplicações em domínios de dados complexos, as operações dependentes da relação de ordem total não são aplicáveis, pois nem todos os dados complexos são naturalmente ordenáveis. Mesmo as operações de igualdade são de pouca utilidade, uma vez que a probabilidade de dois elementos indexados serem exatamente iguais é muito pequena. Assim, as comparações por similaridade são mais naturais e é o fator mais importante nas consultas. Por exemplo, em aplicações médicas, consultas como "encontre as cinco tomografias mais parecidas com esta" podem ser úteis ao diagnóstico de patologias, e em Agrometeorologia, consultas como "encontre os meses em que as temperaturas médias são mais parecidas com as atuais" podem auxiliar na descoberta de padrões para prever comportamentos climáticos. Tais consultas são conhecidas como **Consultas Baseadas em Conteúdo** (*Content Based Queries*) e são usualmente implementadas por métodos de acesso métricos. A semelhança entre dois elementos é determinada por uma função de distância, também chamada métrica, comparando-se os elementos diretamente ou comparando as características extraídas dos mesmos.

A maioria dos métodos de acesso métrico busca diminuir dois fatores: o número de cálculos de distância e o número de acessos a disco para recuperar os elementos, na tentativa de atingir estruturas mais eficientes para retornar elementos com base em seu conteúdo. Ambos fatores podem ser reduzidos eliminando-se sobreposições nodais na estrutura do índice.

Todos os MAM utilizam apenas as propriedades básicas do espaço métrico, buscando um melhor desempenho ajustando o particionamento do espaço e como as propriedades são aplicadas. A Teoria dos Espaços Métricos oferece uma gama de propriedades adicionais que podem auxiliar a melhorar os MAM existentes, possibilitando a construção de novos MAM mais eficientes.

Capítulo

3

Espaços métricos

3.1 Introdução

Os MAM, que vêm sendo desenvolvidos desde o trabalho inicial de Burkhard e Keller [Burkhard & Keller, 1973], utilizam apenas as propriedades fundamentais de uma métrica para melhorar o desempenho de consultas sobre os dados indexados.

Este trabalho visa explorar a Teoria dos Espaços Métricos para identificar outras oportunidades de melhora dos MAM, utilizando o amplo cabedal de recursos teóricos já desenvolvidos pela Matemática discreta. Assim, este capítulo apresenta alguns conceitos e definições importantes sobre os espaços métricos, baseados no livro de Lima [Lima, 1975], tais como espaços normados e aplicações lipschitzianas, que podem ser úteis para isso.

3.2 Espaços métricos

Um espaço métrico é definido como $\mathbb{M} =< \mathbb{S}, d >$, onde $\mathbb{S}$ é um domínio de elementos, e d é a função de distância (ou métrica), definida sobre os elementos em $\mathbb{S}$. A função $d : \mathbb{S}$ x $\mathbb{S} \rightarrow \mathbb{R}^+$ indica a distância entre dois elementos do domínio. Quanto menor esta distância, mais semelhantes eles são e quanto maior a distância, menos semelhantes eles são.

Definição 3.1 *Espaço Métrico. O par* $< \mathbb{S}, d >$ *é chamado espaço métrico sempre que* d *atender as seguintes condições:*

1. $d(s_1, s_1) = 0$; *(identidade)*
2. $0 < d(s_1, s_2) < \infty$, $s_1 \neq s_2$; *(não negatividade)*
3. $d(s_1, s_2) = d(s_2, s_1)$; *(simetria)*
4. $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2)$. *(desigualdade triangular)*

onde $s_1, s_2, s_3 \in \mathbb{S}$.

As propriedades 1 e 2 garantem a identidade entre elementos iguais e a não negatividade da distância entre dois elementos diferentes. A propriedade 3 afirma que a distância $d(s_1, s_2)$ é uma função simétrica das variáveis s_1 e s_2. A propriedade 4 chama-se desigualdade triangular (Figura 3.1), e tem origem no fato de que, no plano euclidiano, o comprimento de um dos lados de um triângulo não excede a soma dos outros dois.

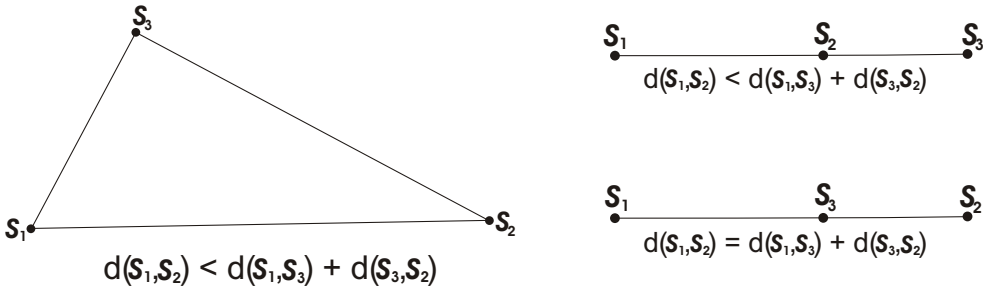


Figura 3.1: Condição da desigualdade triangular.

Assim, um espaço métrico é um par $(\mathbb{S}, d)$, onde $\mathbb{S}$ é um conjunto e d é uma métrica em $\mathbb{S}$. Os elementos de distintos espaços métricos podem ser de natureza bem distinta, como números, vetores, matrizes, funções, conjuntos, etc. Mas todos elementos são sempre chamados de *pontos* de $\mathbb{S}$.

Para efeito de utilização em um SGBD, um subconjunto $S \subseteq \mathbb{S}$ representa o conjunto de elementos indexados nos quais as consultas são efetuadas.

3.2.1 Espaços adimensionais

Espaços adimensionais são espaços onde os elementos não possuem uma dimensão fixa, ou seja, seus elementos são adimensionais. O conjunto de palavras é um exemplo deste tipo de espaço, onde cada palavra tem um número variável de caracteres. Para este conjunto, as métricas mais conhecidas são as de **Levenshtein**, em especial a métrica L_{Edit} [Levenshtein, 1966].

A métrica L_{Edit} retorna o número mínimo de operações de edições (inserções, remoções e substituições de caracteres) necessárias para transformar um cadeia de caracteres c_1 numa outra cadeia c_2. Uma característica desta métrica é que a distância mínima é zero e a máxima pode ser o tamanho da maior palavra do domínio. Essa distância máxima em muitas línguas e dicionários reais dificilmente representa um valor grande. Nesse sentido, a discriminação tende a ser pequena. Outras aplicações demandam buscas por subpalavras, como no caso de bases de seqüências genômicas. Porém nesse caso, a aplicação da L_{Edit} é limitada, levando ao uso de outros tipos de funções mais adequadas a este domínio, como por exemplo a métrica apresentada em [Kahveci & Singh, 2001].

3.2.2 Espaços multidimensionais

O espaço $\mathbb{R}^n$ caracteriza-se por seus dados poderem ser vistos como pontos em um espaço de dimensão n. Por exemplo, se as características visuais de imagens forem representadas por vetores numéricos de n posições, estes vetores terão dimensões fixas para todos os elementos do domínio, as quais também podem ser indexadas por árvores métricas, se uma métrica estiver associada a todo o domínio. Porém, a composição deste vetores podem assumir diferentes domínios de dados para grupos de dimensões, como descritores diferentes de texturas no caso de imagens. Elementos destes conjuntos dizem pertencer a espaços multidimensionais, onde as métricas de **Minkowski** [Wilson & Martinez, 1997] (família L_p definida em 4.2) podem ser utilizadas para comparar os elementos.

3.2.3 Espaços Vetoriais

Definição 3.2 *Espaço Vetorial. Um espaço vetorial é formado por:*

1. *Um conjunto V, cujos elementos são chamados de vetores;*
2. *Um corpo F, cujos elementos são denominados escalares;*
3. *Uma operação $+ : V \times V \rightarrow V$, denominada adição de vetores;*
4. *Uma operação $* : K \times V \rightarrow V$, que define a multiplicação por escalar.*

Diz-se que V é um espaço vetorial sobre K quando as operações $+$ e $*$ satisfazem as seguintes propriedades:

- *Adição*

  1. *Para cada $u, v \in V$, $u + v = v + u$ (comutatividade)*
  2. *Para cada $u, v, w \in V$, $(u + v) + w = u + (v + w)$ (associatividade)*
  3. *Existe um vetor 0, tal que para cada $u \in V, 0 + u = u$ (neutro aditivo)*
  4. *Para cada $u \in V$, existe $-u \in V$ tal que $u + (-u) = 0$ (inverso aditivo)*

- *Multiplicação por escalar*

  1. *Para cada $\alpha \in F$ e cada $u, v \in V$, $\alpha(u + v) = \alpha u + \alpha v$ (distributividade)*
  2. *Para cada $\alpha, \beta \in F$ e cada $u \in V$, $(\alpha + \beta)u = \alpha u + \beta u$ (distributividade)*
  3. *Para cada $\alpha, \beta \in F$ e cada $u \in V$, $(\alpha\beta)u = \alpha(\beta u)$ (associatividade)*
  4. *Para cada $u \in V$, $1u = u$ (neutro multiplicativo)*

3.2.4 Espaços normados

Definição 3.3 *Norma. Seja V um espaço vetorial. Uma norma em V é denotada por $|\ |$ e é uma função que satisfaz as seguintes condições:*

1. *Se $s \neq 0$ então $|s| \neq 0$;*
2. *$|\lambda \cdot s| = |\lambda| \cdot |s|$;*
3. *$|s + t| \leq |s| + |t|$.*

onde $s, t \in V$ e λ é escalar.

Um espaço vetorial normado é um espaço vetorial no qual está definida uma norma. Exemplos de espaços vetoriais normados podem ter normas induzidas pelas métricas L_p, onde, para $s = (s_1, \ldots, s_n) \in \mathbb{R}^n$, tem-se:

$$|_{L_2}s| = \sqrt{\sum (s_i)^2},\ |_{L_1}s| = \sum |s_i|,\ |_{L_\infty}s| = \max |s_i|$$

Um espaço vetorial normado real é um espaço vetorial sobre $\mathbb{R}$ dotado de uma norma. Se V é um espaço vetorial normado, então $d : V \times V \to \mathbb{R}^+$, definida por $d(s,t) = |s-t|$ é uma métrica sobre V, pois:

- $d(s,t) = |s-t| = 0 \iff s-t = 0 \iff s = t$
- $d(s,t) = |s-t| = |(-1)(t-s)| = |-1||t-s| = |t-s| = d(t,s)$
- $d(s,t) = |s-t| = |s-w+w-t| \leq |s-w| + |w-t| = d(s,w) + d(w,t)$

A métrica d assim definida chama-se *métrica induzida pela norma* em V.

Um espaço métrico pode ter uma norma associada, onde a norma pode ser definida como sendo a distância de um elemento para a origem, ou ainda, para qualquer elemento pertencente ao domínio. Assim, teríamos a norma $|s_1| = d(s_1, \Phi)$, onde $s_1 \in S$ *e* $\Phi \in \mathbb{S}$.

3.2.5 Propriedades dos espaços métricos

Definição 3.4 *Produto cartesiano de espaços métricos. Sejam M e N dois espaços métricos, cujas métricas são indicadas por d_M e d_N. O produto cartesiano $M \times N$ é, como conjunto, formado pelos pares ordenados $z = (x,y)$, onde $x \in M$ e $y \in N$. Podemos dotar o produto $M \times N$ de uma métrica, definindo a distância de $z = (x,y)$ a $z' = (x',y')$, como sendo:*

$$d(z,z') = d(x,x') + d(y,y')$$

ou

$$d(z,z') = \sqrt[p]{\sum_{i=1}^{n} |d(x,x') - d(y,y')|^p}, \text{ onde } p \in \mathbb{N}^*$$

Outros exemplos são:

$$d(z,z') = \max\{d(x,x'), d(y,y')\}$$

ou ainda

$$d(z,z') = \sqrt{d(x,x')^2 + d(y,y')^2}.$$

A definição acima é a formalização do que realmente acontece quando concatenamos diferentes vetores de características de elementos na recuperação por conteúdo. As vezes

podemos até concatenar diferentes vetores associados a diferentes métricas, se estendida a definição adequadamente.

Definição 3.5 *Bolas e esferas. Uma bola aberta de centro s_i e raio $r > 0$ é o conjunto $B(s_i; r)$ dos pontos do espaço métrico M cuja distância ao ponto s_i é menor do que r, ou seja:*

$$B(s_i; r) = \{x \in M : d(x, s_i) < r\}.$$

Uma bola fechada é assim definida por:

$$B(s_i; r) = \{x \in M : d(x, s_i) \leq r\}.$$

Uma esfera é formada pela casca do conjunto, sendo definida:

$$S(s_i; r) = \{x \in M : d(x, s_i) = r\}.$$

A definição mais usada pelos MAM é a bola fechada, onde o raio de cobertura de uma bola é definido como a maior distância dos elementos contidos nesta bola ao elemento central, ou seja,

$$r = \max_{s_j \in B(s_i;r)} d(s_i, s_j).$$

O conceito de esfera foi implicitamente utilizado em [Burkhard & Keller, 1973], mais especificamente na primeira técnica proposta neste, conforme visto na Seção 2.2 do Capítulo 2. Note-se também que uma consulta por abrangência pede pela bola fechada $B(s_q, \xi)$.

Definição 3.6 *Hiperplano Generalizado. Sejam dois pontos s_1 e s_2, $s_1 \neq s_2$. Um hiperplano generalizado é o conjunto de pontos Q que satisfazem $d(Q_i, s_1) = d(Q_i, s_2)$. Assim, um ponto x é dito pertencer à partição de s_1 se $d(x, s_1) < d(x, s_2)$.*

O conceito de hiperplano foi utilizado em [Uhlmann, 1991] na criação do MAM *GH-tree*, onde o hiperplano divide o espaço em dois subespaços (partições). A Figura 3.2 mostra um exemplo desta técnica. Note que os elementos {a,c,e} pertencem à partição de s_1, enquanto que os elementos {b,d,f} pertecem à partição de s_2.

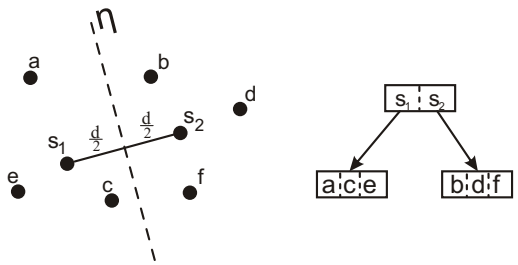


Figura 3.2: Divisão do espaço por um hiperplano.

Definição 3.7 *Métricas equivalentes. Sejam d e d' sobre o mesmo conjunto M. Diz-se que d e d' são métricas equivalentes se, para cada* $p \in M$*, qualquer que seja a bola* $B_d(p, \epsilon)$*, existe* $\lambda > 0$ *de maneira que* $B_{d'}(p, \lambda) \subset B_d(p, \epsilon)$ *e, vice-versa, dada uma bola qualquer* $B_{d'}(d, \epsilon)$ *existe* $B_d(p, \lambda) \subset B_{d'}(p, \epsilon)$

Pode-se trocar uma métrica d por d', desde que estas sejam equivalentes. A ideia de passar de uma métrica para outra mais fina, é a generalização da ideia de tomar uma unidade de medida menor, que resultaria em distâncias maiores. Note que as métricas mais finas não produzem distâncias maiores, mas sim "topologicamente maiores".

O efeito de se tomar uma distância mais fina ou mais grossa resulta, quando da sua aplicação na recuperação por conteúdo, na alteração da seletividade das consultas. Métricas mais finas descartam mais elementos e portanto são mais seletivas. Note-se que nesses sistemas, a troca de métricas equivalentes devem seguir critérios a fim de se evitar falsos negativos [Pola et al., 2006].

Definição 3.8 *Imersão Isométrica. Sejam* M_1 *e* M_2 *espaços métricos. A aplicação* $f : M_1 \to M_2$ *é uma imersão isométrica quando* $d(x, y) = d(f(x), f(y))$ *para qualquer* $x, y \in M_1$*.*

Uma imersão isométrica preserva distâncias.

Definição 3.9 *Isometria. Uma isometria é uma imersão isométrica injetora e sobrejetiva.*

A composição de duas isometrias bem como a inversa de uma isometria são sempre isometrias.

Definição 3.10 *Contrações fracas. São contrações fracas as aplicações* $f : M \to N$ *tais que* $d(f(x), f(y)) \leq d(x, y)$ *para quaisquer* $x, y \in M$*.*

Um exemplo importante de contração fraca é qualquer projeção do produto cartesiano $M = M_1 \times M_2 \times \ldots \times M_n$ usando a métrica L_2, como por exemplo a projeção $p_i : M \to M_i$, pois:

$$d(p_i(x), p_i(y)) = d(x_i, y_i) \leq \sqrt{d(x_1, y_1)^2 + \ldots + d(x_n, y_n)^2} = L_2(x, y).$$

Definição 3.11 *Sejam M_1 e M_2 espaços métricos. Diz-se que a aplicação $f : M_1 \to M_2$ é contínua no ponto $a \in M_1$ quando, para todo $\epsilon > 0$, é possível obter $\delta > 0$, tal que $d(x, a) < \delta$ implica $d(f(x), f(a)) < \epsilon$. Quando a aplicação é contínua em todos os pontos $a \in M_1$, diz-se que ela é contínua.*

É importante notar que a noção de continuidade num ponto é local, isto é, depende do comportamento de f apenas nas proximidades do ponto, o que permite aplicar o conceito para cada centro de consulta, porém não aplicável para o conjunto de dados armazenado como um todo (veja a Figura 3.3).

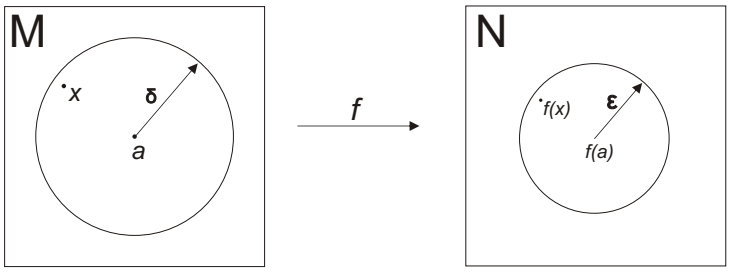


Figura 3.3: Continuidade pontual da aplicação $f : M \to N$.

Definição 3.12 *Dados dois espaços métricos (M, d) e (N, d'), onde d e d' denotam as métricas sobre os conjuntos M e N respectivamente, uma aplicação $f : M \to N$ é chamada de aplicação lipschitziana caso exista uma constante $c > 0$, tal que $d'(f(x), f(y)) \leq c\, d(x, y)$ para todo $x, y \in M$.*

Nesse caso, se f é contínua e derivável em todos os pontos $a \in M$, prova-se que toda função derivada e limitada em um intervalo (o qual pode ser ilimitado) é lipschitziana.

É imediato que se c for constante de Lipschitz, então todo $c' > c$ também será.

O menor valor da constante c é chamada constante de Lipschitz da aplicação f. Se $c = 1$ a aplicação é chamada de mapeamento curto (*short map*), caso $c < 1$ a aplicação é chamada

de contração. Assim, se existe $c \geq 1$ de modo que:

$$\frac{1}{c} d(x,y) \leq d'(f(x), f(y)) \leq c\ d(x,y) \tag{3.1}$$

então f é chamada de aplicação bilipschitz.

Na literatura, intui-se que algumas medidas usadas para modelar a percepção de similaridade humana algumas vezes contradizem em diferentes maneiras os axiomas métricos [Santini & Jain, 1999]. Acredita-se que os axiomas métricos são muito restritivos no contexto de consulta por similaridade. Um dos axiomas mais criticados é a desigualdade triangular, embora talvez seja o axioma mais importante na parte de otimização na área de indexação métrica [Ashby & Perrin, 1988]. Então, constantemente buscam-se meios para que funções de similaridade (que não são métricas) possam operar de maneira semelhante à percepção humana na recuperação baseada em conteúdo. Se for possível identificar uma aplicação lipschitziana que faça determinado subconjunto $S \subset \mathbb{S}$ passar a atender a expressão 3.1, torna-se possível expressar formas de comparação que revelam a percepção humana como métricas em um espaço induzido pela aplicação.

Definição 3.13 *Ângulos em espaços métricos [Valentine, 1975]. Dados três elementos $s_1, s_2, s_3 \in M$, podemos deduzir a partir da propriedade da desigualdade triangular que*

$$-1 \leq \frac{d(s_1,s_2)^2 + d(s_1,s_3)^2 - d(s_2,s_3)^2}{2.d(s_1,s_2).d(s_1,s_3)} \leq 1 \tag{3.2}$$

A partir desta expressão, denota-se o símbolo $\angle s_1 s_2 s_3$ o ângulo com vértice em s_2, determinado pela fórmula

$$\angle s_1 s_2 s_3 = arccos(\frac{d(s_2,s_1)^2 + d(s_2,s_3)^2 - d(s_1,s_3)^2}{2.d(s_2,s_1).d(s_2,s_3)}) \tag{3.3}$$

Para que os ângulos formados no espaço sejam bem definidos é necessário impor no espaço a lei dos cossenos euclidiana. Para isso, considere o exemplo bidimensional da Figura 3.4. Deve-se garantir que $\angle s_1 s_2 s_3 = \angle s_1 s_2 s_4$ sempre que s_4 esteja na reta ligando s_2 a s_3 e também garantir que $\angle s_1 s_2 s_3 = \pi - \angle s_1 s_2 s_5$ sempre que s_5 esteja na reta que liga s_2 a s_3 mas s_1 esteja entre s_3 e s_5.

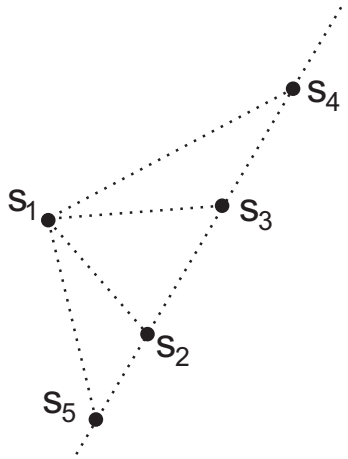


Figura 3.4: Exemplo bidimensional de 5 elementos para checagem de ângulos em um espaço métrico.

3.3 Imersão de espaços métricos no $\mathbb{R}^n$

Muito estudado na área da matemática, a imersão de espaços permite que um conjunto de dados possa ser imergido em outro espaço para que as propriedades do espaço alvo possam ser aplicadas no conjunto de dados. O tipo de imersão tratada aqui é a imersão de espaços métricos em espaços euclidianos de baixa dimensionalidade. Na maioria dos casos, uma série de condições devem ser satisfeitas para que a imersão possa ser feita. Note que enquanto um mapeamento do espaço pode transformar os dados e induzir erros no mapeamento, a imersão é sempre segura pois sempre é controlada.

Uma vez feita a imersão dos dados, onde o termo mais usado na literatura é *embedding*, as propriedades daquele espaço alvo podem ser utilizadas sem qualquer erro associado. No caso da imersão de espaços métricos em espaços euclidianos, todas as operações geométricas são permitidas.

Nesta seção, é mostrada que a lei dos cossenos pode ser utilizada a partir de uma imersão do espaço métrico em um espaço euclidiano bidimensional.

O trabalho de Wilson [Wilson, 1935] resume as condições da imersão de espaços métricos em espaços euclidianos, e mostra que uma condição chamada "convexidade externa" não é necessária para a imersão. Ele conclui seu trabalho definindo um teorema sobre a imersão discutida, onde define as condições necessárias para se realizar uma imersão deste tipo. O teorema enunciado pelo autor é apresentado a seguir.

Theorem 3.3.1 *Quaisquer $n+1$ pontos de um espaço métrico podem ser imersos em $\mathbb{R}^n$ a não ser que exista algum conjunto de $k+3$ pontos, $1 \leq k \leq n-2$, determinando três ângulos ou subespaços angulares $a_1a_2 \ldots a_k : a_ra_s$, $a_1a_2...a_k : a_ra_t$, $a_1a_2...a_k : a_sa_t$, tais que sua soma seja maior que 2π ou a propriedade de desigualdade triangular não seja satisfeita.*

Para exemplificar, considere o exemplo do grafo ilustrado na Figura 3.5. Neste grafo, considere as distâncias entre os elementos como sendo a soma do peso das arestas no caminho mínimo entre eles. Assim, utilizando-se a definição de ângulos definida em 3.13, temos

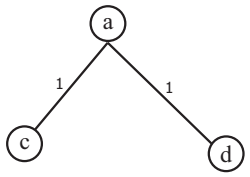


Figura 3.5: Exemplo de um grafo simples para mapeamento em $\mathbb{R}^2$.

$$\angle cad = arccos(-1) = 180$$
$$\angle dca = arccos(1) = 0$$
$$\angle adc = arccos(1) = 0$$

Deste modo, detectamos que os elementos são colineares, centralizados em "a".

Pelo teorema, apesar de que quaisquer três pontos possam ser imersos em $\mathbb{R}^2$, desde que satisfaçam a propriedade de desigualdade triangular, um conjunto com cardinalidade $n > 3$ nem sempre é imersível em $\mathbb{R}^{n-1}$. Por exemplo, considere o grafo mostrado na Figura 3.6, onde novamente a distância entre os elementos é definida como o camínho mínimo entre eles. Este grafo com 4 elementos não é imersível no $\mathbb{R}^3$, pois o espaço de ângulos em a é maior do que 2π. Para ser imersível, teríamos como condição que

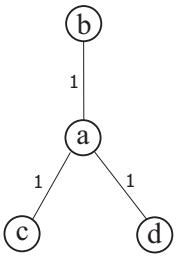


Figura 3.6: Exemplo de um grafo simples que não pode ser mapeado em $\mathbb{R}^3$.

$$\angle bad + \angle cad + \angle bac \leq 360$$

porém

$$\begin{aligned} \angle bad &= arccos((1+1-4)/(2.1.4)) = 180 \\ \angle cad &= arccos((1+1-4)/(2.1.4)) = 180 \\ \angle bac &= arccos((1+1-4)/(2.1.4)) = 180 \end{aligned}$$

então, temos que $\angle bad + \angle cad + \angle bac = 540$. Portanto não imersível em $\mathbb{R}^3$.

Após a imersão do espaço, é possível utilizar as propriedades do espaço imerso, como por exemplo a lei dos cossenos no espaço euclidiano. A lei dos cossenos utiliza a trigonometria para relacionar lados de triângulos com seus ângulos opostos. Assim, em um triângulo qualquer ABC, de lados AB, BC, e CD com medidas c, a e b, respectivamente, podemos relacionar os ângulos e as medidas da seguinte forma

$$\begin{aligned} a^2 &= b^2 + c^2 - 2\ b\ c\ cos\hat{A} \\ b^2 &= a^2 + c^2 - 2\ a\ c\ cos\hat{B} \\ c^2 &= a^2 + b^2 - 2\ a\ b\ cos\hat{C} \end{aligned}$$

3.4 A maldição da alta dimensionalidade

Quando existem muitos graus de liberdade, os dados tendem a se espalhar no espaço. Como consequência, as distâncias entre os elementos próximos e distantes se tornam similares, reduzindo as diferenças das distâncias entre os elementos. De fato, em espaços de alta dimensionalidade, os elementos tendem a ficar longe entre si. Em consultas pelos vizinhos mais próximos, uma vez que o mais próximo foi encontrado, qualquer pequeno aumento no raio de cobertura no raio ativo irá englobar muitos elementos de uma só vez. Este feito degenera os MAM, pois eles não conseguem particionar o espaço adequadamente, aumentando a sobreposição no nó [Volnyansky & Pestov, 2009]. Ambos métodos de acesso, métrico ou espacial, são sensíveis a este problema.

Para verificar este efeito da alta dimensionalidade, é reproduzido aqui o experimento descrito por Katayama e Satoh [Katayama & Satoh, 2001], illustrando o efeito do aumento da dimensionalidade na distribuição das distâncias entre elementos do conjunto.

O experimento consiste em criar vários conjuntos de dados com a mesma cardinalidade, tendo cada um dimensionalidade maior que o anterior, e mensurar a distância mínima e máxima entre os pares de elementos de cada conjunto. Cada elemento foi gerado aleatoriamente em uma distribuição uniforme dentro de uma faixa de valores $[0, 1)$ em cada dimensão. Então, cada conjunto criado corresponde a um conjunto de pontos em um hiper-cubo unitário

com dimensão do conjunto associado.

Foram gerados conjuntos com 100,000 pontos usando 2, 4, 8, 16, 32, 64 e 128 dimensões. A Figura 3.7(a) mostra as distâncias mínima, máxima (diâmetro do conjunto) e média entre cada par de pontos do conjunto. Como pode ser visto, a distância mínima entre os elementos do conjunto de dimensão 128 é maior que a metade do diâmetro mensurado do mesmo conjunto.

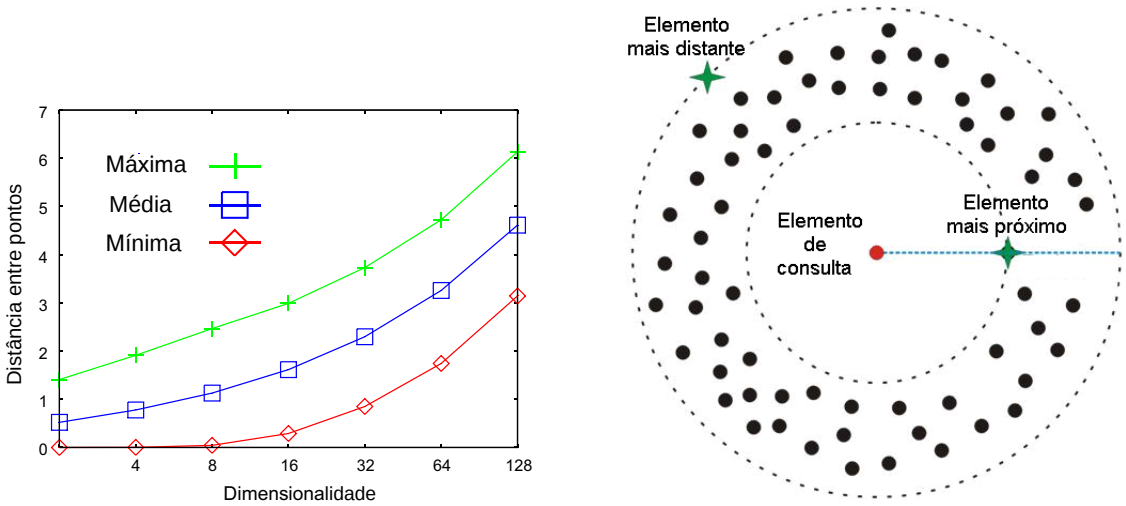


Figura 3.7: Efeitos da maldição da dimensionalidade. (a) Distância mínima, máxima e média entre pontos imersos em hiper-cubos unitários de dimensionalidade variada. (b) Efeito ilustrativo da maldição da dimensionalidade em consultas por similaridade.

O efeito é equivalente a posicionar cada elemento perto da borda do espaço do conjunto, assim nenhum elemento tem realmente um vizinho mais próximo. Este efeito pode ser visto na Figura 3.7(b). Note a distância mínima entre os elementos e o elemento de consulta tende a possuir um grande valor. A execução de consultas por similaridade nestes espaços é muito custosa, e índices levam a ter desempenho similar ou pior que a busca sequencial.

3.5 Conclusão

A Teoria dos Espaços Métricos é ideal para tratar consultas por similaridade, pois estas são operações naturais a dados imersos em tais espaços, além dos axiomas métricos que são usados como mecanismos para a poda de cálculos de distância desnecessários. Baseando-se em uma função de distância métrica e em suas propriedades, é possível elaborar técnicas de indexação eficientes, capazes de responder a essas consultas em tempo hábil. Porém, o

espaço métrico possui propriedades não exploradas na área de recuperação por conteúdo, que podem contribuir para métodos de acesso mais abrangentes e mais eficientes.

Capítulo

4

Funções de distância e suas correlações

4.1 Introdução

Embora na Matemática exista uma diferença de definição sobre os termos função de distância e métrica, nesta tese consideramos que o termo função de distância mede a dissimilaridade entre pares de elementos e é considerada uma métrica, onde a propriedade de desigualdade triangular é atendida. Neste capítulo serão mostradas diversas funções de distâncias e suas características, como limites de valores e abrangências entre elas. Também é analisada a troca de funções de distâncias em consultas, onde seu impacto pode causar distorções nas respostas das consultas.

A definição de distância é diferente de similaridade. A distância é uma medida que quanto maior é o valor, mais diferentes os elementos são. Já a similaridade, quanto maior é o valor, mais semelhantes, ou seja, mais similares os elementos são. Portanto, quanto maior é a distância, menor é a similaridade.

Este capítulo considera a notação: $s_1 = \{s_{11}, s_{12}, \ldots s_{1n}\}$ e $s_2 = \{s_{21}, s_{22}, \ldots s_{2n}\}$ para indicar dois vetores de dimensionalidade n, com $s_1, s_2 \in S \subset \mathbb{R}^n$.

4.2 Distância Minkowski

A distância Minkowski é a função de distância mais utilizada para dados vetoriais, e sua grande vantagem é ser independente da origem do espaço do conjunto de dados, sendo assim invariante a operações de translação. Ela é definida como

$$d(s_1, s_2) = \sqrt[p]{\sum_{i=1}^{n} |s_{1i} - s_{2i}|^p} \ . \tag{4.1}$$

Existem três casos especiais desta função de distância que são frequentemente utilizados. No caso em que $p = 1$ (L_1) ela é chamada de distância entre quarteirões de cidades (*City Block*, Manhattan), correspondendo ao somatório do módulo das diferenças entre as coordenadas. Nesse caso, o conjunto de pontos de mesma distância r forma um losango. Para $p = 2$ (L_2) ela se torna a tradicional distância euclidiana, a qual é invariante a rotação e translação, normalmente usada para distância entre vetores. O conjunto de pontos de mesma distância para L_2 forma uma circunferência. Calculando-se o limite da Equação 4.1 quando p tende ao infinito, obtém-se a métrica (L_∞), também chamada de distância *Chebyshev* ou *Infinity*, na qual o conjunto de pontos com mesma distância formam um quadrado. Para estes casos específicos, a fórmula resultante se reduz às seguintes:

- $L_1(s_1, s_2)$ (*City-Block*): $\sum_{i=1}^{n} |s_{1i} - s_{2i}|$
- $L_2(s, t)$ (*Euclidiana*): $(\sum_{i=1}^{n} |s_{1i} - s_{2i}|^2)^{1/2}$
- $L_\infty(s, t)$ (*Chebyshev*): $max_{i=1}^{n} |s_{1i} - s_{2i}|$

Uma propriedade importante sobre estas métricas é dada pela correlação entre elas pela seguinte desigualdade:

$$L_\infty(s, t) \leq L_2(s, t) \leq L_1(s, t) \leq n \cdot L_\infty(s, t) \tag{4.2}$$

que mostra que o grau de seletividade das métricas muda conforme a distribuição das distâncias dada por cada métrica.

A Figura 4.1 apresenta as formas geométricas geradas pelas funções L_1, L_2 e L_∞ para um espaço bidimensional. Uma generalização para um espaço n-dimensional fará com que

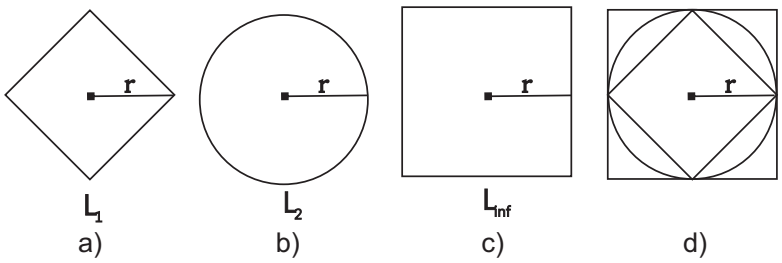


Figura 4.1: Formas geométricas geradas de acordo com a métrica L_p utilizada.

a função L_1 gere um hiper-octaedro de dimensão n, L_2 gere uma hiper-esfera de dimensão n, e assim por diante. Na Figura 4.1(d), a sobreposição das representações geométricas das funções sugere que diferentes métricas englobam subespaços diferentes, como expresso pela desigualdade 4.2.

A similaridade total, ou seja, a igualdade dos elementos, ocorre quando $d(s_1, s_2) = 0$, e a dissimilaridade total ocorre quando $d(s_1, s_2) \to \infty$

4.3 Distância Canberra

A distância Canberra atribui peso na diferença dos pares de coordenadas em cada dimensão comparada. Ela é definida como

$$d(s_1, s_2) = \sum_{i=1}^{n} \frac{|s_{1i} - s_{2i}|}{|s_{1i}| + |s_{2i}|} \quad . \tag{4.3}$$

Cada termo da somatória assume um valor entre zero e um. Se uma coordenada é zero, o termo se torna unitário qualquer que seja o valor da outra coordenada, não permitindo a interferência numérica de uma grande disparidade em uma dada dimensão.

Os valores da distância Canberra se modificam caso o conjunto de dados seja transladado, e ela é muito sensível a pequenas mudanças quando as coordenadas são próximas de zero. Caso um par de coordenadas comparadas possuam valor zero, define-se o valor zero para a distância.

A similaridade total ocorre quando $d(s_1, s_2) = 0$, e a dissimilaridade total ocorre quando $d(s_1, s_2) = n$.

4.4 Distância Bray-Curtis

A distância Bray-Curtis é também chamada distância Sorensen. Ela utiliza o método de normalização e é usualmente aplicada em botânica, ecologia e ciências ambientais, onde os valores das coordenadas são sempre não negativos.

Essa função de distância analiza o espaço como uma grade, de maneira similar à distância entre blocos de cidades (L_1). Ela é definida como

$$d(s_1, s_2) = \frac{\sum_{i=1}^{n} |s_{1i} - s_{2i}|}{\sum_{i=1}^{n} (|s_{1i}| + |s_{2i}|)} \ , \tag{4.4}$$

ou de maneira equivalente

$$d(s_1, s_2) = \frac{\sum_{i=1}^{n} |s_{1i} - s_{2i}|}{\sum_{i=1}^{n} |s_{1i}| + \sum_{i=1}^{n} |s_{2i}|} \ . \tag{4.5}$$

A função de distância Bray-Curtis possui a propriedade de que se todas coordenadas são positivas, seu valor está no intervalo de zero a um. Caso um par de coordenadas comparadas possuem valor zero, define-se o valor zero para a distância.

A similaridade total ocorre quando $d(s_1, s_2) = 0$, e a dissimilaridade total ocorre quando $d(s_1, s_2) = n$.

4.5 Separação angular ou distância dos cossenos

O coeficiente de separação angular calcula o cosseno do ângulo entre dois vetores, centrados na origem do conjunto de dados (isto é, centrados no elemento que possui em todas as coordenadas o valor zero, $s_{1i} = 0$). O coeficiente de separação angular é definido como

$$c(s_1, s_2) = \frac{\sum_{i=1}^{n} (s_{1i} \cdot s_{2i})}{\sqrt{\sum_{i=1}^{n} s_{1i}^2 \cdot \sum_{i=1}^{n} s_{2i}^2}} \ . \tag{4.6}$$

O valor do coeficiente angular está no intervalo $[-1, 1]$, como ocorre com os valores dos

cossenos. Se os valores das coordenadas de pelo menos um dos elementos é zero, considera-se o valor um para a distância. A similaridade total neste coeficiente ocorre quando $c(s_1, s_2) = 1$ e a total dissimilaridade ocorre quando $c(s_1, s_2) = -1$.

É importante notar que o coeficiente mede a similaridade ao invés da distância ou dissimilaridade. Para converter o coeficiente c, dado pela Equação 4.6, em uma medida de função de distância, a seguinte equação pode ser aplicada:

$$d(s_1, s_2) = \frac{1 - c(s_1, s_2)}{2} \tag{4.7}$$

Desta maneira, a similaridade total ocorre quando $d(s_1, s_2) = 0$ e a dissimilaridade total ocorre quando $d(s_1, s_2) = 1$.

4.6 Distância de correlação

O coeficiente de correlação é a correlação de separação angular centrada em um lugar diferente da origem do espaço de dados. Ele mede similaridade ao invés da distância ou dissimilaridade.

Existem duas opções que são usualmente utilizadas para definir o posicionamento do centro do espaço. A primeira considera que todas as dimensões são homogêneas, posicionando o centro no valor médio das coordenadas dos elementos. Assim, o coeficiente de correlação fica definido como

$$c(s_1, s_2) = \frac{\sum_{i=1}^{n} |(s_{1i} - \overline{s_1}) \cdot (s_{2i} - \overline{s_2})|}{\sqrt{\sum_{i=1}^{n} (s_{1i} - \overline{s_1})^2 \cdot \sum_{i=1}^{n} (s_{2i} - \overline{s_2})^2}} \ , \tag{4.8}$$

onde $\overline{s_1} = \sum_{i=1}^{n} s_{1i}$ e $\overline{s_2} = \sum_{i=1}^{n} s_{2i}$

A segunda opção é posicionar no centro de cada *cluster* ao qual os elementos pertencem, novamente assumindo que o centro é o ponto onde as coordenadas possuem o valor médio

das coordenadas dos elementos do *cluster*. Assim, o coeficiente fica definido como

$$c(s_1, s_2) = \frac{\sum_{i=1}^{n} \left|(s_{1i} - \overline{S_i}) \cdot (s_{2i} - \overline{S_i})\right|}{\sqrt{\sum_{i=1}^{n} (s_{1i} - \overline{S_i})^2 \cdot \sum_{i=1}^{n} (s_{2i} - \overline{S_i})^2}} \ , \tag{4.9}$$

onde $\overline{S_i} = \sum_{t \in S} s_{ti}$.

Esta opção é frequentemente aplicada para calcular distâncias quando o conjunto de dados é dinâmico, pois a distância entre os elementos muda dependendo dos elementos que são inseridos no conjunto (ou nos *clusters*).

Se algum dos elementos corresponder ao centro escolhido, o coeficiente de correlação nesta comparação é definida com valor um. O valor do coeficiente de correlação está no intervalo $[-1, 1]$. Novamente, para converter o coeficiente a uma medida de distância, a Equação 4.7 pode ser aplicada. A similaridade total ocorre quando $d(s_1, s_2) = 0$ e a dissimilaridade total ocorre quando $d(s_1, s_2) = 1$.

4.7 Distância quadrática

A distância quadrática é definida como a medida de dissimilaridade entre dois vetores s_1 e s_2 da mesma distribuição.

$$d(s_1, s_2) = [(s_1 - s_2)^T W (s_1 - s_2)]^{1/q} = \left[\sum_{j=1}^{n} \left(\sum_{i=1}^{n} (s_{1i} - s_{2i}) w_{ij}\right) (s_{1j} - s_{2j})\right]^{1/q} \ , \tag{4.10}$$

onde $W = [w_{ij}]$ é uma matrix de pesos $n \times n$ definida positiva.

Caso a matriz de pesos seja definida como a matriz identidade, a distância quadrática se torna a distância euclidiana. Caso seja definida diagonal, então a medida resultante da distância é chamada de distância euclidiana normalizada:

$$d(s_1, s_2) = \sqrt{\sum_{i=1}^{n} \frac{(s_{1i} - s_{2i})^2}{\sigma_i^2}} \ , \tag{4.11}$$

onde σ_i^2 é o desvio padrão da i-ésima dimensão do conjunto de dados.

A similaridade total ocorre quando $d(s_1, s_2) = 0$ e a dissimilaridade total tende a $d(s_1, s_2) = \infty$.

4.8 Distância Mahalanobis

A função de distância Mahalanobis é definida como a função de distância quadrática com a matriz de pesos definida como a inversa da matriz de covariância V of $A_1 \ldots A_n$, onde A_i é o vetor de pesos para a dimensão i:

$$d(s_1, s_2) = [\det V]^{1/n} (s_1 - s_2)^T V^{-1} (s_1 - s_2) \ . \tag{4.12}$$

Assim como a distância quadrática, a similaridade total ocorre quando $d(s_1, s_2) = 0$ e a dissimilaridade total tende a $d(s_1, s_2) = \infty$.

4.9 Distância de edição

A distância de edição [Levenshtein, 1966], também conhecida como distância de Levenshtein, ou simplesmente L_{Edit}, retorna o número mínimo de operações de edições (inserções, remoções e substituições de caracteres) necessárias para transformar um cadeia de caracteres c_1 numa outra cadeia c_2. O algoritmo para determinar esta distância é baseado em programação dinâmica, e usa uma matriz para realizar os cálculos. O algoritmo tem ordem de complexidade $\Theta(m \times n)$, onde m e n são os tamanhos das duas cadeias de caracteres comparadas.

O Algoritmo 1 descreve como calcular a distância de edição entre duas cadeias de caracteres s e t. Uma matriz de ordem $(m + 1 \times n + 1)$ é utilizada para mensurar as operações de edição. No final do procedimento, o resultado está localizado na célula da matriz com posição (m, n). A Tabela 4.1 mostra exemplos de matrizes geradas pelo algoritmo quando comparam-se algumas palavras.

Uma característica desta métrica é que a distância mínima é zero e a máxima é o tamanho da maior palavra do domínio. Existem algumas variações desta métrica, que diferem em quais operações de edição são permitidas. Por exemplo, o custo pode ser considerado

Algoritmo 1: Função de distância de edição.

$L_{Edit}(string\ s, string\ t)$
Entrada: s,t: cadeias de caracteres.
Saída: $dist$: o número de edições para transformar s em t

```
1:  declare a matriz de inteiros D (0..m, 0..n)
2:  para i = 0 to m faça
3:      D(i, 0) = i
4:  fim para
5:  para j = 0 to n faça
6:      D(0, j) = j
7:  fim para
8:  para j = 0 to n faça
9:      para i = 0 to m faça
10:         se t[j] == s[i] então
11:             D(i, j) = D(i-1, j-1)
12:         senão {calcular o mínimo}
13:             {entre as operações   deleção   inserção   substituição   }
14:             D(i, j) = minimum( D(i-1, j) + 1,  D(i, j-1) + 1,  D(i-1, j-1) + 1 )
15:         fim se
16:     fim para
17: fim para
    return D(m,n)
```

diferente dependendo do caracter envolvido em cada comparação, ou pode-se atribuir pesos diferentes nas operações de edição, desde que sejam pesos normalizados para manter a função métrica. Outros exemplos de variações: o tamanho da maior subsequência comum [Bergroth et al., 2000], admitindo apenas inserções e deleções; a distância Damerau-Levenshtein [Damerau, 1964] permitindo inserções, deleções, substituições e a transposição de dois caracteres adjacentes, com a motivação original para ser utilizada em corretores de texto; a distância de Hamming [Hamming, 1986] que só admite substituições, limitando as palavras comparadas a terem o mesmo número de caracteres, e sua extensão, a distância de Hamming generalizada [Bookstein et al., 2002]. Uma propriedade interessante é que se as palavras tiverem o mesmo número de caracteres, a distância de Hamming é um limite superior da distância de Levenshtein.

(a)

		g	a	t	o
	0	1	2	3	4
p	1	1	2	3	4
a	2	2	1	2	3
t	3	2	2	1	3
o	4	4	3	2	**1**

(b)

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	**3**

(c)

		S	a	t	u	r	d	a	y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	**3**

Tabela 4.1: Exemplos de matrizes geradas durante o cálculo da distância de edição.

4.10 Outras distâncias

Assim como a distância Mahalanobis, outras funções de distância interessantes, baseadas na estatística do conjunto, foram propostas. A maioria delas assume que os vetores de características s_1 e s_2 sejam distribuições.

A divergência de Kullback-Leibler (KL) é considerada uma medida da extensão nas quais duas funções de densidade de probabilidade correspondem, e é expressa da seguinte maneira.

$$d(s_1, s_2) = -\int p(x) ln \frac{\tilde{p}(x)}{p(x)} dx \ , \tag{4.13}$$

Pode ser mostrado que $d \geq 0$. Para duas distribuições discretas a integração se torna a somatória nos bins de s_1 e s_2.

A divergência de Jeffrey é a versão simétrica da divergência de Kullback-Leibler, com respeito ás funções de densidade de probabilidade $p(x)$ e $\tilde{p}(x)$. Ela é definida como

$$d(s_1, s_2) = \sum_{i=1}^{n} \left| s_{1i} \ln \frac{s_{1i}}{m_i} + s_{2i} \ln \frac{s_{2i}}{m_i} \right| \ , \tag{4.14}$$

onde $m_i = \frac{s_{1i}+s_{2i}}{2}$

A medida estatística χ^2 mede o fator da probabilidade que uma distribuição seja escrita a partir de outra, sendo definida como

$$d_{\chi^2}(s_1, s_2) = \sum_{i=1}^{n} \frac{s_{1i} - m_i}{s_{2i}} \ , \tag{4.15}$$

onde $m_i = \frac{s_{1i}+s_{2i}}{2}$

A distância Bhattacharyya mede a separabilidade estatística de classes espectrais, levando a uma estimativa da probabilidade de uma classificação correta, possuindo interpretação geométrica. Ela é definida como

$$d(s_1, s_2) = \sqrt{1 - \sum_{i=1}^{n} \sqrt{s_{1i} * s_{2i}}} \ . \tag{4.16}$$

A distância de "movimento de terra" (*Earth Mover's Distance* - EMD) considera cada

valor de uma distribuição como uma quantidade a ser movida (terra) para outra distribuição (os buracos). A EMD é definida como o custo mínimo necessário para transferir uma distribuição para outra. A vantagem da EMD é que ela pode medir a distância em distribuições com diferentes números de *bins*. Ela foi projetada originalmente para ser usada para tratar problemas de transporte [Hitchcock, 1941] mas também pode ser usada para mensurar a similaridade de imagens [Rubner et al., 1998].

A Tabela 4.2 sumariza algumas das funções aqui descritas mostrando os limites inferiores e superiores das funções. O estudo das funções de distância vai muito além de conhecer todas as variantes, e visa responder ao desafio de comparar melhor novos tipos de dados complexos.

4.11 Valores limites de funções de distância

Um dos estudos que auxiliam na troca ou escolha das diferentes funções de distância é o estudo da abrangência delas. Em algumas funções de distância, a determinação da propriedade de limite inferior (*lower bound*) é garantida se a função é homogênea no espaço das características. Por exemplo, a família de funções Minkowski é homogênea para qualquer $p \geq 1$ dado. Mas para outras funções de distância, a homogeneidade é perdida quando o centro de consulta é mudado ou quando o raio da consulta é alterado.

Para algumas funções de distância pode-se analizar o conjunto de dados para descobrir se uma função de distância é limitada inferiormente por outra. Por exemplo, se for desejado descobrir qual a relação de limites entre as funções de distância Canberra e a Minkowski, devemos descobrir se a seguinte relação é verdadeira, aplicando-a no conjunto:

$$\sum_{i=1}^{n} \frac{|s_{1i} - s_{2i}|}{|s_{1i}| + |s_{2i}|} \geq \left(\sum_{i=1}^{n} |s_{1i} - s_{2i}|^p \right)^{1/p}$$

mas se $p = 1$, queremos determinar se

$$\sum_{i=1}^{n} \frac{|s_{1i} - s_{2i}|}{|s_{1i}| + |s_{2i}|} - |s_{1i} - s_{2i}| \geq 0$$

resultando,

$$\sum_{i=1}^{n} \frac{|s_{1i} - s_{2i}|(1 - (|s_{1i}| + |s_{2i}|))}{|s_{1i}| + |s_{2i}|} \geq 0, \forall s_1, s_2 \in S$$

e uma vez que $L_1 > L_2 > \ldots > L_\infty$, a relação vale para qualquer $p \geq 1$.

Porém, caso queira-se encontrar uma constante para definir os limites entre as funções, podemos fazer isso majorando uma das distâncias. Por exemplo, caso majoremos a distância Canberra d_c:

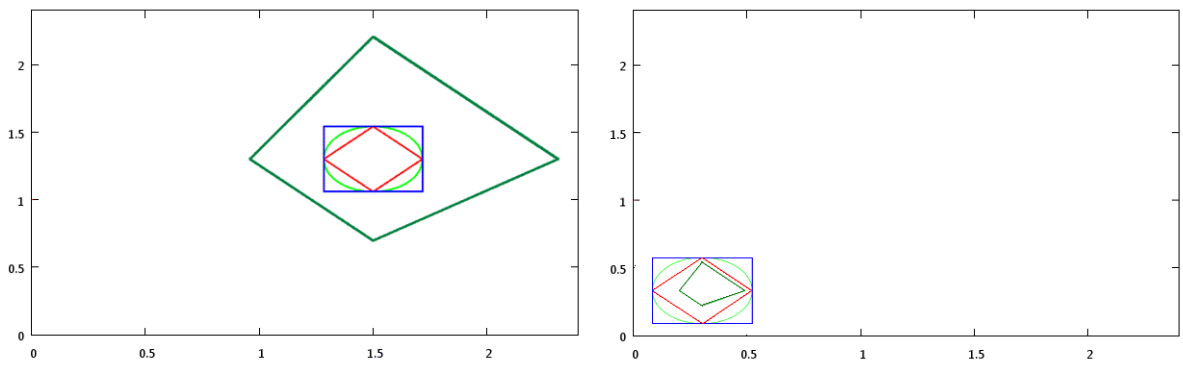
$$d_c \geq \sum_{i=1}^{n} \frac{|s_{1i} - s_{2i}|}{c} = \frac{1}{c} \sum_{i=1}^{n} |s_{1i} - s_{2i}| = \frac{1}{c}\ d_{L_1}$$

tem-se, uma vez que $d_{L_1} > d_{L_2} > \ldots > d_{L_{inf}}$,

$$d_c \geq \frac{1}{c}\ d_{L_p}$$

onde $c = max(|s_{1i}| + |s_{2i}|), \forall s_1, s_2 \in S$

Assim, podemos determinar matematicamente qual função de distância é limitada inferiormente por outra, e em quais condições isso ocorre, de acordo com o conjunto de dados que estamos utilizando e também com o centro de consulta. Algumas visualizações destas funções podem ser vistas na Figura 4.2, onde o centro de consulta é deslocado em cada caso, mudando-se assim a abrangência da função Canberra em relação à Minkowski. Note enquanto que na Figura 4.2(a) a distância Canberra é mais abrangente que as Minkowski, a situação é inversa ao se aproximar da origem do espaço, como visto na Figura 4.2(b).



(a) (b)

Figura 4.2: Variação da abrangência das funções de distância pelo centro de consulta.

Pode-se obter uma relação de comparação para as métricas L_p também. Sabe-se que elas são homogêneas no espaço das características, e portanto translações não modificam a abrangência entre elas. Como já mencionado, o limite inferior destas funções é bem definido como

$$L_\infty \leq \ldots \leq L_2 \leq L_1 \leq n\ L_\infty$$

onde n é o número de dimensões do espaço de características.

Para obter um relacionamento dinâmico entre estas métricas considerando-se a propriedade de limite inferior, pode-se utilizar a inequação de média generalizada (*generalized mean inequality*), que é definida como

$$M(\alpha) = \left(\frac{x_1^\alpha + x_2^\alpha + \ldots + x_n^\alpha}{n} \right)^{1/\alpha}$$
$$\alpha \leq \beta \Longrightarrow M(\alpha) \leq M(\beta)$$

Desenvolvendo-se a equação anterior, tem-se

$$\frac{(x_1^\alpha + x_2^\alpha + \ldots + x_n^\alpha)^{1/\alpha}}{n^{1/\alpha}} \leq \frac{(x_1^\beta + x_2^\beta + \ldots + x_n^\beta)^{1/\beta}}{n^{1/\beta}}$$

Finalmente,

$$(x_1^\alpha + x_2^\alpha + \ldots + x_n^\alpha)^{1/\alpha} \leq \frac{n^{1/\alpha}}{n^{1/\beta}}\ (x_1^\beta + x_2^\beta + \ldots + x_n^\beta)^{1/\beta}$$

onde $\alpha \leq \beta$.

Os relacionamentos entre as métricas L_p podem ser obtidos associando-se diferentes valores para α e β. Por exemplo, caso queira-se comparar o relacionamento entre as métricas L_p e L_{p+1}, tem-se

$$L_{p+1} \leq L_p \leq \frac{n^{1/p}}{n^{1/(p+1)}}\ L_{p+1}$$

onde $1 \leq p \leq \infty$.

4.12 Substituições de funções de distância

Dentre as várias funções de distância definidas, podem existir aplicações que necessitam utilizá-las de modo intermitente, para aumentar ou reduzir a seletividade da consulta conforme desejado.

O uso das propriedades dos espaços métricos em MAMs deve ser muito bem analizado, pois em consultas, a troca de uma função de distância que retorna um conjunto A por outra mais abrangente que retorna um conjunto resposta B, não é possível garantir que $A \subset B$. Isso ocorre porque muitos MAM armazenam distâncias pré-calculadas, e a inequação triangular falha quando misturam-se funções de distância. Mesmo que o limite superior seja atendido em alguns casos, a propriedade utilizada para poda nas estruturas é o limite inferior da inequação, ou seja, seja $s_p, s_i, s_q \in S$

$$\underbrace{|d(s_i, s_p) - d(s_p, s_q)|}_{limite\ inferior} \leq d(s_q, s_i) \leq \underbrace{d(s_i, s_p) + d(s_p, s_q)}_{limite\ superior}$$

Considere a seguinte situação. Uma *Slim-tree*, é construída com uma métrica, e deseja-se utilizar uma outra métrica para consultá-la. Uma consulta feita com a métrica L_1 retorna um conjunto A, mas se realizarmos uma consulta com a métrica L_2 produzindo B, não é possível garantir que $A \subset B$. O fato se agrava mais pois se outra *Slim-tree* for construída com a métrica L_2 e a mesma consulta for executada produzindo o resultado C, não se pode garantir que $C = B$. Essa divergência de resultados é dada pela falha na inequação triangular, explicada a seguir.

Considere a Figura 4.3, onde um MAM *Slim-tree* é construído utilizando-se a métrica L_1. Neste caso a distância armazenada na estrutura corresponde à distância $d(s_i, s_p)$, e caso a função de distância utilizada na consulta mude, as distâncias $d(s_q, s_p)$ e $d(s_q, s_i)$ são alteradas.

A Figura 4.4 mostra gráficos das desigualdades triangulares para este exemplo, para um conjunto aleatório de pontos. Foram testadas diversas triangulações, e o limite superior da propriedade da desigualdade triangular foi verificada mudando-se a métrica de consulta. Nas figuras, a curva contínua representa a distância armazenada na estrutura $L_2(s_q, s_i)$,

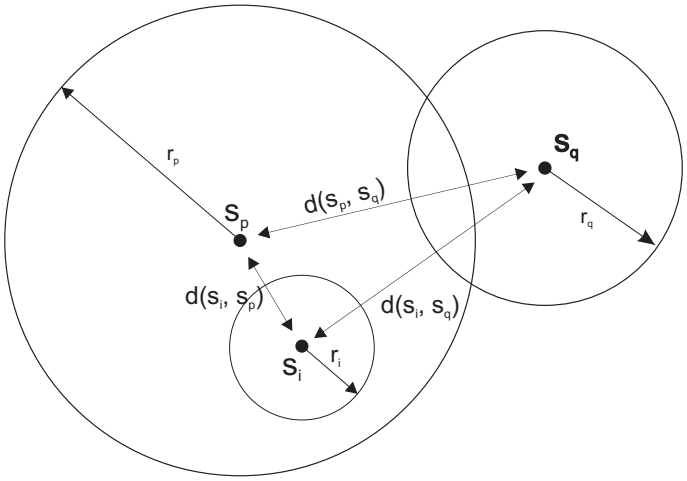
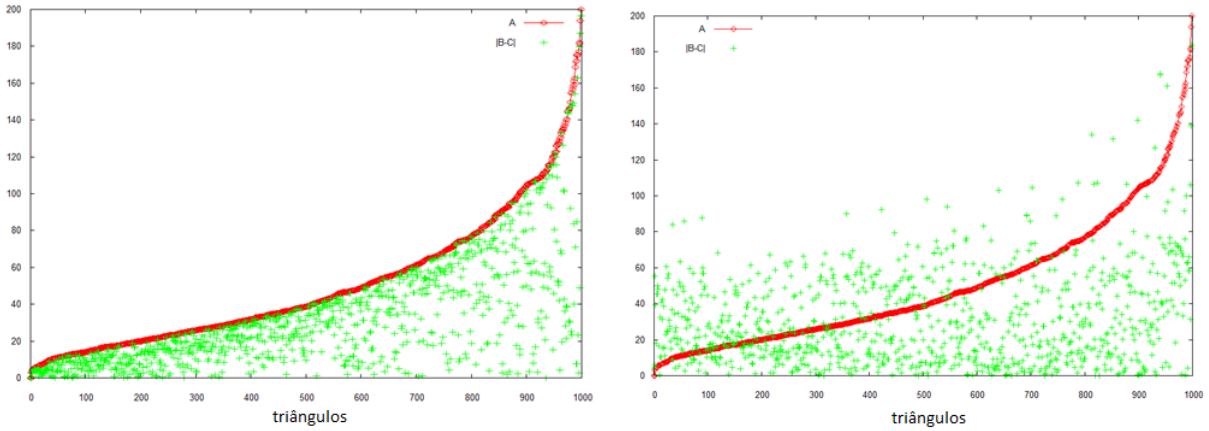


Figura 4.3: Modelo de consulta em um nó de uma *Slim-tree*.

enquanto os pontos representam os valores dos limites superiores, definidos como $|d_{L_2}(s_i, s_p) - d_{L_2}(s_p, s_q)|$ na Figura 4.4(a), e $||d_{L_1}(s_i, s_p) - d_{L_2}(s_p, s_q)|$ na Figura 4.4(b). A Figura 4.4(a) mostra que o limite superior é preservado, portanto a desigualdade triangular vale para a métrica L_2 usada como consulta. Já na 4.4(b) o limite superior não é preservado, pois utilizou-se uma métrica diferente (L_1) da utilizada na construção da estrutura.



(a) $L_2(s_q, s_i) \geq |L_2(s_i, s_p) - L_2(s_p, s_q)|$ (b) $L_2(s_q, s_i) \geq |L_1(s_i, s_p) - L_2(s_p, s_q)|$

Figura 4.4: Verificação da propriedade da desigualdade triangular para dois casos de consulta.

4.13 Conclusão

Existem muita funções de distância desenvolvidas e seu uso deve ser avaliado de acordo com as características dos dados que se deseja comparar. Apesar de muitas delas poderem ser derivadas de outras, cada uma tem uma propriedade diferente, na qual pode-se tirar proveito para melhorar a qualidade da recuperação dos dados.

A abrangência de funções mostra quão genéricas elas são, e uma vez que se identifique uma correlação entre elas, pode-se mensurar quando uma função será mais abrangente que a outra, aumentando assim a generalidade da consulta conforme o perfil do usuário em questão.

$s_1 = \{s_{11}, s_{12}, \ldots s_{1n}\}, \quad s_2 = \{s_{21}, s_{22}, \ldots s_{2n}\}, \quad s_1, s_2 \in S$

$\overline{s_1} = \sum_{i=1}^{n} s_{1i}, \quad \overline{s_2} = \sum_{i=1}^{n} s_{2i}$ $\qquad W =$ matriz de pesos

$\overline{S_i} = \sum_{t \in S} s_{ti}$ $\qquad V =$ matriz de covariância

Funções			Limites
Minkowski $L_p(s_1, s_2) = \sqrt[p]{\sum_{i=1}^{n} \lvert s_{1i} - s_{2i} \rvert^p}$			[0,∞]
Manhattan $L_1(s_1, s_2) = \sum_{i=1}^{n} \lvert s_{1i} - s_{2i} \rvert$	Euclidiana $L_2(s_1, s_2) = \sqrt[2]{\sum_{i=1}^{n} (s_{1i} - s_{2i})}$	Infinity $L_\infty(s_1, s_2) = \max_{i=1}^{n} \lvert s_{1i} - s_{2i} \rvert$	[0,∞]
Canberra $\text{Canberrad}(s_1, s_2) = \sum_{i=1}^{n} \frac{\lvert s_{1i} - s_{2i} \rvert}{\lvert s_{1i} \rvert + \lvert s_{2i} \rvert}$			[0,n]
Bray-Curtis $\text{BCd}(s_1, s_2) = \frac{\sum_{i=1}^{n} \lvert s_{1i} - s_{2i} \rvert}{\sum_{i=1}^{n} (\lvert s_{1i} \rvert + \lvert s_{2i} \rvert)}$			[0,n]
Cossenos $\cos(s_1, s_2) = 1 - \frac{\sum_{i=1}^{n} (s_{1i} \cdot s_{2i})}{2\sqrt{\sum_{i=1}^{n} s_{1i}^2 \cdot \sum_{i=1}^{n} s_{2i}^2}}$			[0,1]
Correlação $\text{Correld}(s_1, s_2) = 1 - \frac{\sum_{i=1}^{n} \lvert (s_{1i} - \overline{s_1}) \cdot (s_{2i} - \overline{s_2}) \rvert}{2\sqrt{\sum_{i=1}^{n} (s_{1i} - \overline{s_1})^2 \cdot \sum_{i=1}^{n} (s_{2i} - \overline{s_2})^2}}$ ou $\text{Correld}(s_1, s_2) = 1 - \frac{\sum_{i=1}^{n} \left\lvert (s_{1i} - \overline{S_i}) \cdot (s_{2i} - \overline{S_i}) \right\rvert}{2\sqrt{\sum_{i=1}^{n} (s_{1i} - \overline{S_i})^2 \cdot \sum_{i=1}^{n} (s_{2i} - \overline{S_i})^2}}$			[0,1]
Quadrática $\text{Quadraticd}(s_1, s_2) = (s_1 - s_2)^T W (s_1 - s_2) = \sum_{j=1}^{n} \left(\sum_{i=1}^{n} (s_{1i} - s_{2i}) w_{ij} \right) (s_{1j} - s_{2j})$			[0,∞]
Mahalanobis $\text{Mahalanobisd}(s_1, s_2) = [\det V]^{1/n} (s_1 - s_2)^T V^{-1} (s_1 - s_2)$			[0,∞]

Tabela 4.2: Funções de distância e seus limites de valores

Capítulo

5

Técnicas de imersões de espaços

5.1 Introdução

Este capítulo mostra como a imersão de espaços pode ser utilizada para melhorar o desempenho de métodos de acesso métrico. Primeiramente, é mostrado um método de acesso em memória, a *MM-tree*, desenvolvido neste trabalho. Após a apresentação desta estrutura, será mostrado como utilizar as condições necessárias para realizar um mapeamento de espaços métricos no espaço $\mathbb{R}^n$ de forma exata, ou seja, sem perda de correlação de distâncias. Este mapeamento permite aplicar propriedades pertinentes ao espaço mapeado diretamente na estrutura de dados. Estas propriedades são utilizadas para refinar o particionamento do espaço, otimizando o acesso na *MM-tree*, gerando assim uma família de estruturas.

5.2 *MM-tree*

Muitas aplicações necessitam que o SGBD construa rapidamente índices em dados que possam ocupar a memória principal. Com o aumento da capacidade da memória principal e seu custo cada vez mais baixo, vem sendo útil indexar dados na memória principal, principalmente quando o índice necessita ser freqüentemente reconstruído. Os MAMs baseados em disco forçam os nós a terem muitos elementos, excluindo boas abordagens para o particionamento do espaço de dados. Mais ainda, cada MAM baseado em disco tenta minimizar ao

máximo o número de acessos a disco, porque este acesso é muito lento comparado ao acesso à memória principal e isto afeta o desempenho das estruturas.

A *MM-tree* (*Main Memory Metric-tree*) [Pola et al., 2007] foi desenvolvida visando atender à necessidade de responder rapidamente as consultas por similaridade sobre conjuntos de dados S em espaços métricos $\mathbb{S}$ que possam ser mantidos em memória.

A ideia geral da *MM-tree* é selecionar dois elementos $s_1, s_2 \in S$ como pivôs para particionar o espaco em quatro regiões disjuntas. Cada elemento s_i é associado a uma única partição, então não há sobreposição nodal para consultas pontuais. O particionamento das regiões é baseado nas distâncias dos elementos aos pivôs como indicado na Tabela 5.1, onde r é a distância entre os pivôs, ou seja, $r = d(s_1, s_2)$.

O mesmo procedimento é recursivamente aplicado a cada região, criando a árvore. Note que são necessários somente dois cálculos de distâncias por nível da árvore para determinar qual é a região adequada.

$\mathbf{d(s_i, s_1)\ \theta\ r}$	$\mathbf{d(s_i, s_2)\ \theta\ r}$	**Região**
$<$	$<$	I
$<$	$\geq$	II
$\geq$	$<$	III
$\geq$	$\geq$	IV

Tabela 5.1: Regiões do espaço onde o elemento s_i deve ser associado na *MM-tree*.

Cada nó de uma *MM-tree* tem a mesma estrutura, que é definida como segue:

$$no = [s_1, s_2, d(s_1, s_2), Ptr_1, Ptr_2, Ptr_3, Ptr_4]$$

onde s_1 e s_2 são os pivôs, $d(s_1, s_2)$ é a distância entre os pivôs, e Ptr_1 a Ptr_4 são os ponteiros para as quatro subárvores que armazenam os elementos nas regiões de 1 a 4, respectivamente.

A Figura 5.1 mostra uma representação gráfica da estrutura do primeiro nível de uma *MM-tree* indexando oito elementos: {a,b,c,d,e,f,g,h}, sendo (a,b) os pivôs. Note-se que a distância $d(a, b) = r$ define o raio da bola de cada pivô, formando as regiões disjuntas I, II, III e IV. A *MM-tree* é construída num esquema *top down*, onde cada nó contém até dois elementos, particionando o subespaço sempre de acordo com a mesma regra.

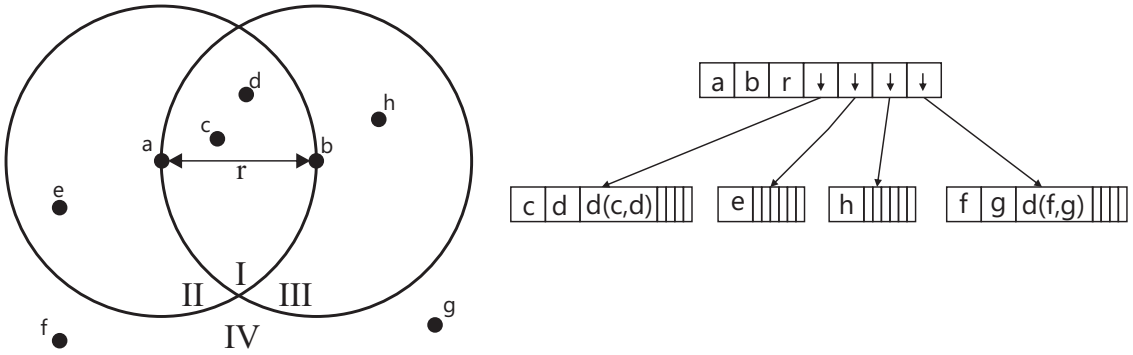


Figura 5.1: Um exemplo da *MM-tree* indexando 8 elementos.

5.2.1 Construção

A *MM-tree* é uma árvore dinâmica baseada em memória, então novos elementos podem ser inseridos na árvore a qualquer momento. Para inserir um novo elemento s_n, o algoritmo navega na árvore começando pela raiz, realizando uma consulta pontual, procurando pelo nó apropriado para armazenar o novo elemento. A cada nó, s_n é comparado aos pivôs (s_1, s_2) e a região correta é determinada baseando-se na Tabela 5.1. Se o elemento a ser inserido não estiver presente na árvore, ele é inserido num nó folha, caso contrário ele é descartado para evitar repetições na estrutura. O Algoritmo 2 descreve como deve ser feita a inserção na estrutura.

Controle de balanceamento

A árvore criada pela inserção de novos elementos sem qualquer controle gera árvores não balanceadas em altura. Deixado sem controle, isso pode comprometer as operações de busca. Para evitar que a árvore deteriore, esta seção apresenta uma técnica para melhorar a construção da estrutura da *MM-tree*: o algoritmo de semi-balanceamento que controla o desbalanceamento dos nós folhas.

Para ilustrar a técnica, será usado aqui um exemplo gráfico da técnica, mostrando na Figura 5.2 o que ocorre se o elemento j tiver que ser incluído na árvore original da Figura 5.1. A Figura 5.2(a) mostra a árvore antes do balanceamento, e a Figura 5.2(b) mostra a árvore após o balanceamento. Note que os pivôs (a, b) na 5.2(a) são trocados por (e, d) para que o elemento j possa ser reposicionado, evitando-se que a árvore cresça um nível.

A técnica é descrita da seguinte maneira. Note-se que em uma subárvore *MM-tree* com

Algoritmo 2: Inserção de um novo elemento na *MM-tree*

Insere(s_n, raiz N)

Entrada: s_n: elemento a ser inserido; N: raiz da subárvore

```
Seja N a raiz, s1 and s2 os pivôs de N e r a distância d(s1, s2).
se N não está cheio então
   Insira sn em N
senão
   d1 = d(s1, sn)
   d2 = d(s2, sn)
   se d1 < r então
      se d2 < r então
         N = região I
      senão
         N = região II
      fim se
   senão
      se d2 < r então
         N = região III
      senão
         N = região IV
      fim se
   fim se
   Insere(sn, N)
fim se
```

dois níveis, pode-se armazenar até 10 elementos (2 elementos por nó, contando 5 nós no total, incluindo a raiz da subárvore). Considerando a árvore da Figura 5.2(a), nota-se que um nó descendente da raiz cobrindo a região I contendo {c,d} já tem 2 elementos, portanto está cheio e qualquer inserção neste nó causará um aumento no nível da subárvore, o que ocorre com a inserção do elemento j, indicado em cor cinza na figura, embora as 10 posições da subárvore não estejam todas ocupadas.

O algoritmo de semi-balanceamento (mostrado como Algoritmo 3) busca re-eleger os

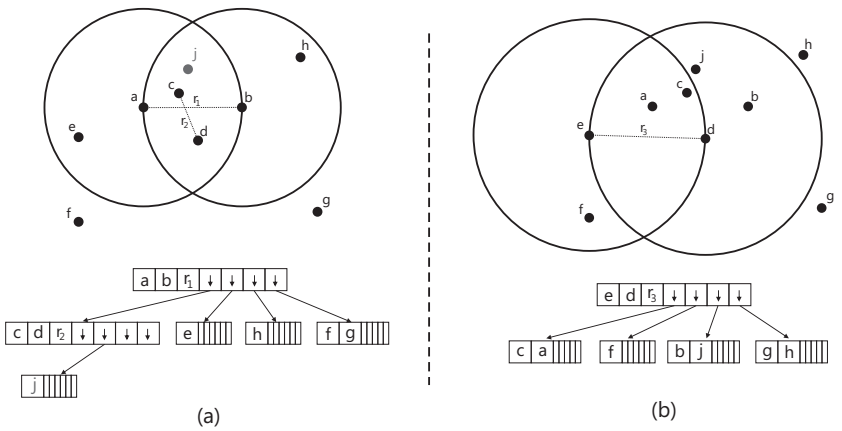


Figura 5.2: Um exemplo de aplicação do algoritmo de semi-balanceamento.

pivôs no nó pai quando seu nó filho (que é um nó folha) é alvo de uma inserção e está cheio, mas algum de seus irmãos não. Esta mudança de pivôs pode rearranjar os elementos em uma maneira que todos os elementos possam ser armazenados num único nível (linhas 1 a 6). Se nenhuma das combinações de pivôs forem satisfatória, a subárvore ganha um nível (linhas 7 e 8).

Algoritmo 3: Técnica de Semi-balanceamento na *MM-tree*

Semi-balanceamento(N, s_n) Entrada: N: ponteiro para a raiz da subárvore; s_n: o novo elemento.
Saída: N: ponteiro para a nova subárvore com s_n inserido.

```
1: Seja O uma lista de elementos filhos de N adicionando s_n e os pivôs de N.
2: Faça NN ser um novo nó.
3: para todo s_i ∈ O faça
4:     para cada s_j ∈ O, s_j ≠ s_i faça
5:         NN.esvazie()
6:         NN.adicionePivos(s_i, s_j)
7:         Faça El ser a lista dos elementos composta por (O − s_i − s_j)
8:         NN.insereElementos(El)
9:         se elementos em El foram distribuídos no mesmo nível sob N então
10:            Retorne a nova raiz NN.
11:        fim se
12:    fim para
13: fim para
14: N.insereElemento(s_n).
15: Retorne o nó N.
```

Este procedimento é aplicado somente no último nível de nós acima das folhas para evitar a necessidade de se reconstruir toda árvore a cada inserção. Por razões estatísticas, este algoritmo é somente executado quando sua probabilidade de sucesso é alta, ou seja, quando uma subárvore (raiz + nós folha) contiver armazenados até 8 elementos. Esse valor foi obtido empiricamente avaliando grande quantidade de conjuntos de dados.

5.2.2 Consultas

Além de responder a consultas pontuais, a *MM-tree* deve responder às duas consultas mais conhecidas: *range queries* (consultas por abrangência) - *Rq* e *k nearest neighbors queries* (consulta aos vizinhos mais próximos) - $kNNq$. Esses algoritmos se baseiam em determinar quais regiões sobrepõem a região de consulta. O raio ativo para as *Rq* é fixo, enquanto o raio ativo das $kNNq$ é um valor dinâmico que começa no maior valor possível e se reduz conforme o algoritmo avança e encontra vizinhos sucessivamente mais próximos.

O Algoritmo 4 responde uma consulta do tipo *range query* na *MM-tree*. Ele recebe um raio de consulta r_q e o elemento de consulta s_q. Em cada nó visitado, se a distância de s_q aos pivôs for menor que r_q, então aquele pivô é adicionado na resposta (linhas 2 e 3). A seguir, verifica se há intersecções da bola $B(s_q; r_q)$ com cada uma das 4 regiões definidas em cada nó (linhas 4 e 5), visitando as regiões que sobreponham a bola de consulta (linha 6), baseando-se na distância de s_q aos pivôs $s_1 e s_2$ segundo a Tabela 5.2.

Algoritmo 4: Consulta por abrangência na *MM-tree*

RangeQuery (s_q, r_q, $raiz$)

Entrada: s_q: centro de consulta; r_q: raio de consulta; $raiz$: pointero para a subávore.
Saída: *Lista*: lista com a resposta.

1: Calcule as distâncias $d_1 = d(s_1, s_q)$ and $d_2 = d(s_2, s_q)$ no nó atual *root*
2: **se** $d_1 \leq r_q$ **então** $Lista.adicione(s_1)$
3: **se** $d_2 \leq r_q$ **então** $Lista.adicione(s_2)$
4: **para** cada região R_j em (I, II, III, IV) de $raiz$ **faça**
5: **se** o raio de consulta sobrepõe a região apontada por R_j **então**
6: RangeQuery(s_q, r_q, R_j)
7: **fim se**
8: **fim para**

Região I	$(d(s_q, s_2) < r_q + r) \ \wedge \ (d(s_q, s_1) < r_q + r)$
Região II	$(d(s_q, s_2) + r_q \geq r) \ \wedge \ (d(s_q, s_1) < r_q + r)$
Região III	$(d(s_q, s_2) < r_q + r) \ \wedge \ (d(s_q, s_1) + r_q \geq r)$
Região IV	$(d(s_q, s_2) + r_q \geq r) \ \wedge \ (d(s_q, s_1) + r_q \geq r)$

Tabela 5.2: Condições de sobreposição na *MM-tree*.

O Algoritmo 5 responde a uma consulta $kNNq$ na *MM-tree*. Ele mantém uma lista ordenada dos k elementos mais próximos a s_q. Quando o algoritmo inicia, a lista está vazia e o raio ativo r_q tem valor ∞. O raio ativo é reduzido toda vez que a lista é atualizada (quando um elemento mais próximo a s_q é encontrado), tornando-se a maior distância entre s_q e cada um dos elementos da lista. Uma sobreposição entre as regiões ocorre se alguma das condições descritas na Tabela 5.2 são satisfeitas, agora sendo r_q o raio ativo dinâmico. As linhas 15 a 18 testam estas condições e visitam os nó inferiores adequadamente. Note-se que se s_q pertence à região P, esta região deverá ser visitada sem qualquer condição de poda ser testada.

Algoritmo 5: Consulta aos k vizinhos mais próximos na *MM-tree*.

NearestQuery (s_q, k, $raiz$)

Entrada: s_q: centro de consulta; k: número de vizinhos mais próximos; $raiz$: ponteiro para a subárvore.
Saída: $Lista$: lista com a resposta.

```
1: Calcule as distâncias d_1 = d(s_1, s_q) e d_2 = d(s_2, s_q) no nó atual raiz
2: se List.size() < k então
3:     r_q = ∞
4: senão
5:     r_q = Lista[k].distância
6: fim se
7: para i = 1..2 faça
8:     se d_i < r_q então
9:         Lista.adicione(s_i)
10:        Lista.ordene()
11:        Lista.EliminaApos(k)
12:    fim se
13: fim para
14: Seja P a região a qual s_q pertence.
15: NearestQuery(s_q, k, P)
16: se nó tem dois pivôs então
17:    para cada região R_j em (I, II, III, IV) de raiz, R_j ≠ P faça
18:        se bola de consulta intercepta R_j então
19:            NearestQuery(s_q, k, R_j)
20:        fim se
21:    fim para
22: fim se
```

Técnica guiada

O raio ativo no algoritmo $kNNq$ vai sendo reduzido conforme elementos mais próximos a s_q vão sendo encontrados. Conforme este raio se torna menor, mais subárvores podem ser podadas, reduzindo os cálculos de distância. Então, diferentes ordens de pesquisa nas regiões cobertas pela bola de consulta podem levar a podas de subárvores mais efetivamente.

O algoritmo guiado de $kNNq$ escolhe uma sequência de pesquisa nos nós da *MM-tree*, que visa reduzir o raio ativo de consulta mais rapidamente. O algoritmo primeiro visita a região a que o elemento de consulta pertence, visitando as outras regiões de acordo com a sequência descrita na Tabela 5.3, onde d é a distância entre os pivôs.

5.3 O hiperplano métrico

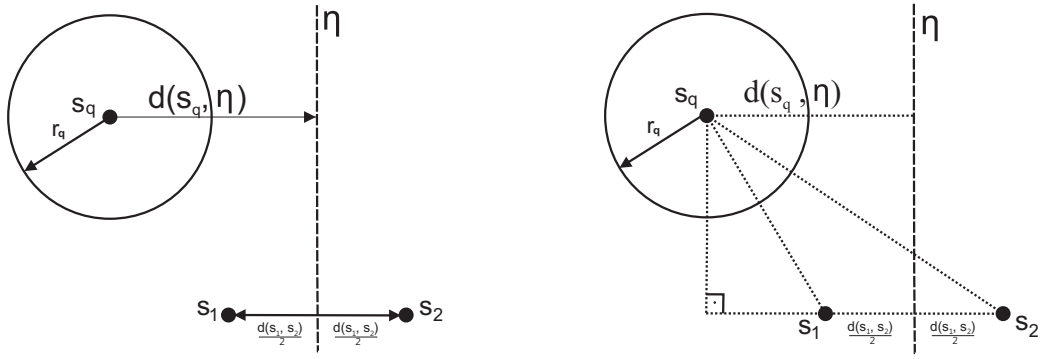
O hiperplano generalizado definido na Seção 3.2.5 particiona o espaço métrico em dois subespaços, a partir de dois pontos posicionados como pivôs. O problema desta técnica é

		ordem de visita	
região onde s_q pertence	condição C	C é verdadeiro	C é falso
I	$d_1 \leq d_2$	I→II→(III,IV)	I→III→(II,IV)
II	$d_2 - d \leq d - d_1$	II→I→IV→III	II→IV→IV→II
III	$d_1 - d \leq d - d_2$	III→I→IV→II	III→IV→I→II
IV	$d_1 \leq d_2$	IV→II→I→III	IV→III→I→II

Tabela 5.3: Sequência de visita nas regiões no algoritmo guiado de $kNNq$.

que, no espaço métrico, não é possível definir a distância de um elemento ao hiperplano generalizado, ou seja, não existe uma métrica ou técnica para calcular essa distância. Consultas como $kNNq$ e Rq ficam dependentes de projeções no eixo dos pivôs para obter uma estimativa de distância.

Na Figura 5.3(a) podemos visualizar a utilidade desta distância, já que desejamos obter a distância $d(s_q, \eta)$ para detectar a intersecção entre a bola de consulta $\mathrm{B}(s_q, r_q)$ e o hiperplano η. Caso a bola de consulta intercepte o hiperplano, ou seja, $r_q \geq d(s_q, \eta)$, a consulta deve analizar a partição de s_2, caso contrário a partição é desconsiderada para busca, caracterizando-se uma poda.



(a) Distância necessária para detecção de intersecção com um hiperplano.

(b) Triangulação para aplicação da lei dos cossenos

Figura 5.3: Imersão e triangulação dos elementos para aplicação da lei dos cossenos.

Sem realizar um mapeamento para um espaço dimensional, não é possível determinar a distância $d(s_q, \eta)$ no espaço métrico, pois o hiperplano η não é um elemento, mas uma definição. Aqui apresentamos uma técnica para calcular esta distância de modo exato, tornando o hiperplano métrico. A técnica consiste em fazer um mapeamento do espaço métrico para o $\mathbb{R}^n$ e aplicar a lei dos cossenos para calcular esta distância, a fim de se obter

as intersecções das bolas de consulta com o hiperplano métrico.

Especificamente, a técnica considera dois pontos como pivôs, definindo um hiperplano generalizado. Assim, para calcular a distância de um terceiro ponto ao hiperplano, um mapeamento destes três pontos para o $\mathbb{R}^2$ é realizado, e a lei dos cossenos é aplicada para calcular a distância. Observe que o mapeamento é exato, de acordo com o Teorema 3.3.1 enunciado na Seção 3.3. Assim, ainda na Figura 5.3(a), se fizermos ligações entre os elementos, formaremos os triângulos indicados como na Figura 5.3(b). Desta maneira, com as triangulações feitas e o espaço mapeado para o $\mathbb{R}^2$, podemos seguramente utilizar as propriedades deste espaço. Para aplicar a lei dos cossenos, considere a Figura 5.4, onde são definidos os valores dos segmentos e a projeção de s_q em $\bar{s_q}$, definindo assim todos triângulos necessários. O que queremos determinar é a distância $X = d(s_q, \eta)$. Então, considerando os triângulos $\{s_q,\ \bar{s_q},\ s_1\}$ e $\{s_q,\ \bar{s_q},\ s_2\}$, podemos aplicar a lei dos cossenos e montar o seguinte sistema:

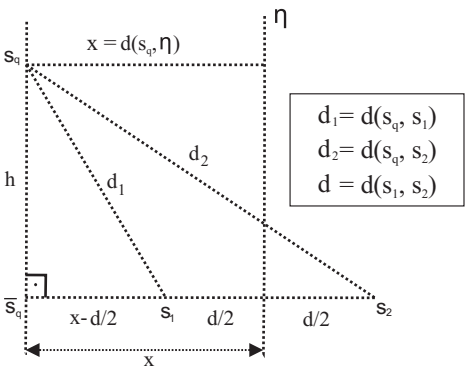


Figura 5.4: Definição das medidas dos segmentos na triangularização.

$$\begin{cases} d_1^2 &= h^2 + (x - d/2)^2 \\ d_2^2 &= h^2 + (x + d/2)^2 \end{cases} \tag{5.1}$$

O sistema possui duas equações e duas váriáveis, portanto com solução única e determinável. Resolvendo este sistema temos que

$$x = \frac{d_2^2 - d_1^2}{2d}$$

Por simetria, podemos generalizar esta fórmula para abranger os casos em que s_q está à esquerda de η ($d_1 \leq d_2$) e à direita de η ($d_2 \leq d_1$), obtendo a fórmula final:

$$d_\eta = d(s_q, \eta) = \frac{|d_1^2 - d_2^2|}{2d} \tag{5.2}$$

Por meio desta técnica definimos o conceito de hiperplano métrico, pois possibilita o

cálculo das distâncias dos elementos do conjunto para um hiperplano generalizado definido a partir de dois elementos pivôs. O hiperplano métrico pode então ser utilizado em MAMs para melhorar o particionamento do espaço e assim diminuir o número de cálculos de distância necessários para realizar as consultas por similaridade.

5.4 A árvore binária métrica - *MB-tree*

A árvore binária métrica (*MB-tree* - *Metric Binary tree*) usa o hiperplano métrico para dividir o espaço métrico em duas regiões, chamadas de partições, como mostrado na Figura 5.5. Os elementos são distribuidos de acordo com as partições às quais eles pertencem, de acordo com a distância deles aos pivôs. Assim, se um elemento s_n está mais próximo de s_1 do que de s_2, ele pertence à partição de s_1. O empate das distâncias, ou seja, quando um elemento está exatamente sobre o hiperplano métrico, é resolvido assumindo-se uma das partições como acolhedoras. Experimentos realizados serviram serviram de base para decidir que a criação de um terceiro ramo para este caso de empate não é necessário, uma vez que para espaços onde o contradomínio da métrica é contínuo, a chance de elementos estarem sobre o hiperplano é muito pequena. Deste modo, nós temos um esquema de indexação clássico de árvore binária no espaço métrico.

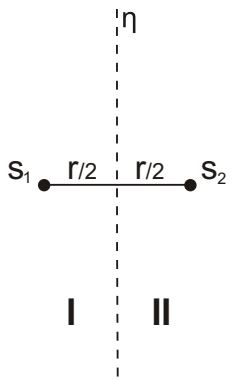


Figura 5.5: Particionamento da árvore binária métrica.

A *MB-tree* tem uma estrutura equivalente à *GH-tree*, porém sua construção é feita de maneira incremental. Assim, por um lado a árvore resultante não é necessariamente balaceada, mas por outro, o custo de construção é muito menor. Nosso interesse aqui é que essa árvore permite explorar o cálculo das distâncias entre bolas e hiperplanos, habilitando que

consultas por abrangência e por vizinhos mais próximos sejam respondidas eficientemente. Lembramos que não foi encontrado registro de como resolver essas consultas sobre a *GH-tree*.

Cada nó da *MB-tree* tem dois elementos e o mesmo esquema de particionamento é utilizado em cada nó, criando-se a árvore hierarquicamente. A vantagem desta estrutura é que ela é simples mas poderosa na divisão do espaço. Todos os nós da *MB-tree* têm a mesma estrutura, definida como:

$$nó : [s_1, s_2, Ptr_I, Ptr_{II}]$$

onde s_1 e s_2 são os pivôs, Ptr_I e Ptr_{II} são os ponteiros para subárvores.

A Figura 5.6 mostra uma representação gráfica do primeiro nível da *MB-tree* indexando seis elementos em duas dimensões: {a,b,c,d,e,f}, escolhendo-se o par (a,b) como pivôs. Note que os elementos são armazenados na árvore de acordo com a partição à qual eles pertencem, necessitando-se de dois cálculos de distância em cada nó.

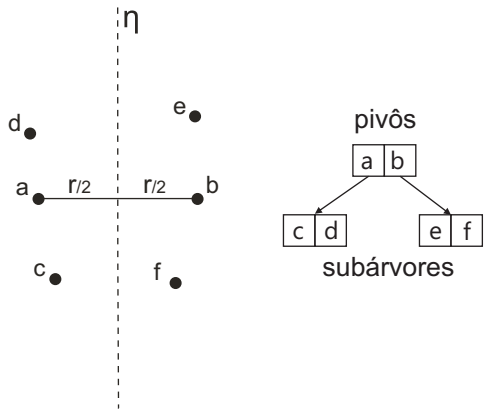


Figura 5.6: Um exemplo de uma árvore binária métrica indexando seis elementos.

Como cada nó da *MB-tree* comporta dois elementos como pivôs, em cada nó necessitamos de dois cálculos de distância para definir a partição adequada ao novo elemento. A próxima subseção explica o processo de inserção na *MB-tree*.

5.4.1 Construção

A *MB-tree* é uma estrutura dinâmica em memória e novos elementos são inseridos na árvore a qualquer momento. Para inserir um novo elemento s_n na estrutura, o algoritmo de inserção navega na estrutura realizando uma consulta pontual, procurando por um nó adequado para

armazenar o novo elemento. Em cada nó, s_n é comparado aos pivôs (s_1, s_2) e a partição adequada é determinada de acordo com as distâncias calculadas.

O procedimento de inserção na *MB-tree* é mostrado no Algoritmo 6. O algoritmo escolhe recursivamente a partição adequada para o novo elemento s_n comparando-o com os pivôs de cada nó.

Algoritmo 6: Inserção na Árvore Binária Métrica

Insere (s_n, raiz N)
Entrada: s_n: elemento a ser inserido; N: raiz da subárvore

```
Sejam s1 e s2 os pivôs do nó N.
se N não está cheio então
   Insira o elemento sn no nó N
senão
   d1 = d(s1, sn)
   d2 = d(s2, sn)
   se d1 < d2 então
      N = partição I
   senão
      N = partição II
   fim se
   Insere(sn, N)
fim se
```

5.4.2 Consulta

Com a técnica de mapeamento, as consultas sobre árvores *MB-tree* podem ser realizadas de modo eficiente, através da utilização da lei dos cossenos. A Figura 5.7 mostra um exemplo de consulta de abrangência para a árvore binária métrica. Aplicando o conceito já visto para calcular a distância do elemento de consulta ao hiperplano que divide o espaço, tem-se as condições de poda de cada partição definidas na Tabela 5.4.

Poda a partição de	se valer a condição
s_2	$(d(s_1, s_q) \leq d(s_2, s_q)) \wedge (\ r_q < d_\eta)$
s_1	$(d(s_1, s_q) > d(s_2, s_q)) \wedge (\ r_q < d_\eta)$

Tabela 5.4: Condições de poda na árvore binária métrica

Com estas condições, pode-se definir quando a bola de consulta $B(s_q, r_q)$ intersecta o hiperplano métrico η, possibilitando escolher quais subárvores da estrutura são realmente necessárias para visitação. Deste modo, podemos efetivamente realizar consultas por similaridade com grau binário de poda. O Algoritmo 7 descreve as operações necessárias para

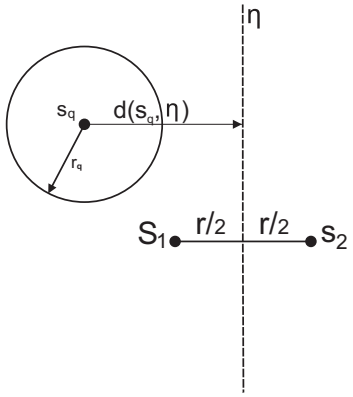


Figura 5.7: Exemplo de consulta por abrangência na árvore binária métrica.

uma consulta por abrangência para a *MB-tree*. A ideia é definir um raio ativo de busca, que pode mudar conforme as regiões forem visitadas e vizinhos mais próximos são atualizados, no caso da consulta aos vizinhos mais próximos. No caso de consultas por abrangência, o raio ativo é fixo e determinado pelo usuário no momento da consulta.

Algoritmo 7: Consulta por abrangência na ABM.

RangeQuery (s_q, r_q, $raiz$)

Entrada: s_q: centro de consulta; r_q: raio de consulta; $raiz$: ponteiro para raiz.
Saída: lista de resultados $Lista$.

1: **se** $raiz ==$ NULL **então**
2: return
3: **fim se**
4: Calcule as distâncias $d_1 = d(s_1, s_q)$, $d_2 = d(s_2, s_q)$ e $d_\eta = \frac{|d_1^2 - d_2^2|}{2d}$
5: **se** $d_1 \leq r_q$ **então** $Lista.adicione(s_1)$
6: **se** $d_2 \leq r_q$ **então** $Lista.adicione(s_2)$
7: **se** root não é um nó folha **então**
8: **se** $d_1 < d_2$ **então**
9: RangeQuery(s_q, r_q, $raiz.ptr_I$)
10: **se** $r_q \geq d_\eta$ **então**
11: RangeQuery(s_q, r_q, $raiz.ptr_{II}$)
12: **fim se**
13: **senão**
14: RangeQuery(s_q, r_q, $raiz.ptr_{II}$)
15: **se** $r_q \geq d_\eta$ **então**
16: RangeQuery(s_q, r_q, $raiz.ptr_I$)
17: **fim se**
18: **fim se**
19: **fim se**

Para a consulta aos vizinhos mais próximos, o procedimento descrito no Algoritmo 8 deve ser utilizado. Note-se que o algoritmo mantém uma lista dos elementos encontrados ordenada pela distância ao centro de consulta s_q, e o raio ativo da bola de consulta é atualizada sempre que um novo elemento é encontrado.

Algoritmo 8: Consulta aos k vizinhos mais próximos na ABM.

NearestQuery (s_q**,** k**,** $raiz$**)** Entrada: s_q: centro de consulta; k: número de vizinhos mais próximos; $raiz$: ponteiro para a raiz.
Saída: lista de resultados $Lista$.

1: **se** $raiz ==$ NULL **então**
2: return
3: **fim se**
4: **se** $Lista.tamanho() < k$ **então**
5: $r_q = \infty$
6: **senão**
7: $r_q = Lista[k].distancia$
8: **fim se**
9: Calcule as distances $d_1 = d(s_1, s_q)$, $d_2 = d(s_2, s_q)$ e $d_\eta = \frac{|d_1^2 - d_2^2|}{2d}$
10: **para** $i = 1..2$ **faça**
11: **se** $d_i < r_q$ **então**
12: $Lista.adicione(s_i)$
13: $Lista.ordene()$
14: $Lista.EliminaApos(k)$
15: **fim se**
16: **fim para**
17: **se** root não é um nó folha **então**
18: **se** $d_1 < d_2$ **então**
19: NearestQuery(s_q, k, $raiz.ptr_I$)
20: **se** $r_q \geq d_\eta$ **então**
21: NearestQuery(s_q, k, $raiz.ptr_{II}$)
22: **fim se**
23: **senão**
24: NearestQuery(s_q, k, $raiz.ptr_{II}$)
25: **se** $r_q \geq d_\eta$ **então**
26: NearestQuery(s_q, k, $raiz.ptr_I$)
27: **fim se**
28: **fim se**
29: **fim se**

5.5 A *MMH-tree*

É possível aumentar o *fanout* da *MM-tree* (Seção 5.2) utilizando-se o hiperplano métrico definido na Seção 5.3. Nesta seção será apresentada uma extensão da *MM-tree*, a *MMH-tree*, onde diferentes combinações utilizando o hiperplano métrico geram diferentes MAMs. A nova estrutura acrescenta o uso do hiperplano métrico para dividir as regiões formadas pelas bolas intersectantes centradas nos pivôs s_1 e s_2, $B_1(s_1; r)$ e $B_2(s_2; r)$ respectivamente, onde $r = d(s_1, s_2)$, como mostra a Figura 5.8. Note-se na figura que o hiperplano métrico η divide as regiões que ele atravessa, formando novas regiões disjuntas. Este mecanismo de divisão das regiões provê a divisão em até seis novas regiões formadas pelas bolas intersectantes juntamente com o hiperplano métrico, indicadas como regiões I, II, III, IV, V e VI. Cada

nó da nova estrutura também tem dois elementos, e o mesmo esquema de particionamento é aplicado em cada nova região, hierarquicamente em um esquema *top-down*. A maior vantagem deste método é prover áreas disjuntas em um nó, onde apenas dois cálculos de distância são necessários para averiguar a correta região de cobertura do elemento, no mesmo tempo em que apenas esses dois cálculos permitem espalhar os elementos em até seis regiões disjuntas.

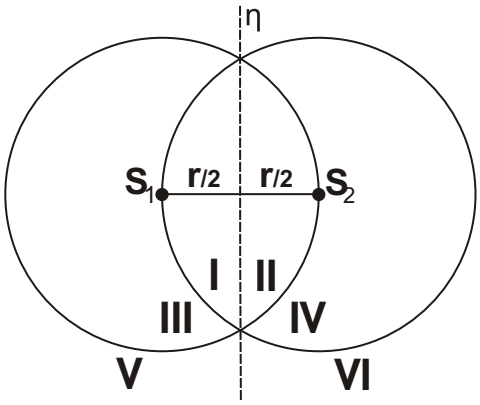


Figura 5.8: Particionamento do espaço utilizando-se o hiperplano métrico.

Como mencionado anteriormente, cada nó da MMH tem a formação semelhante à *MB-tree* e à *MM-tree*, definida como:

$$nó : [s_1, s_2, d(s_1, s_2), Ptr_I, Ptr_{II}, Ptr_{III}, Ptr_{IV}, Ptr_V, Ptr_{VI}]$$

onde s_1 e s_2 são os pivôs, $d(s_1, s_2)$ é a distância entre os pivôs e Ptr_I até Ptr_{VI} são os ponteiros para as seis subárvores. Note que a distância $d(s_1, s_2)$ define o raio das bolas centradas nos pivôs e o hiperplano métrico "corta" o espaço na metade das regiões, formando assim seis regiões disjuntas numeradas de I a VI, que são mostradas na Figura 5.8.

A Figura 5.9 mostra uma representação gráfica do primeiro nível da *MMH-tree* indexando 14 elementos em duas dimensões: {a,b,c,d,e,f,g,h,i,j,k,l,m,n}, tendo o par (a,b) como os pivôs. Pela figura, pode-se notar que cada ramo da árvore representa uma região do espaço formado pela intersecção das bolas e o hiperplano métrico.

Note que cada nó da estrutura MMH comporta até dois elementos e com apenas dois cálculos de distância se pode definir unicamente qual região comporta um dado elemento s_i dentre as seis regiões formadas. A região adequada para um novo elemento s_n é determinada

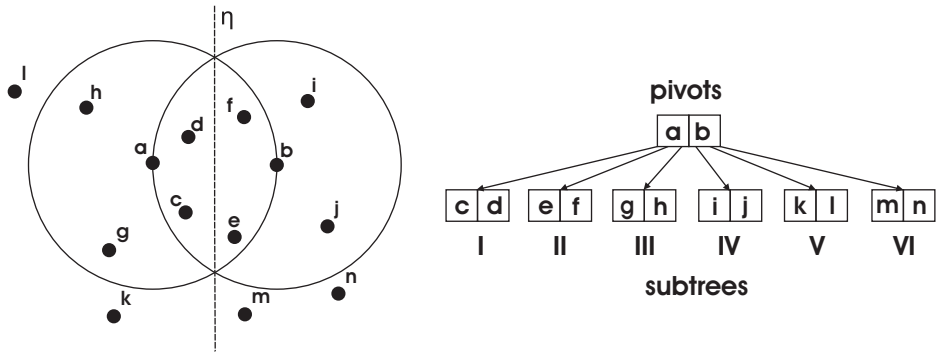


Figura 5.9: Um exemplo da *MMH-tree* indexando 14 elementos.

por um conjunto de condições, definidas na Tabela 5.5. Pela tabela, o símbolo θ pode assumir os operadores $<$ ou $\geq$, formando termos que combinados com outros definem a região de cobertura para um novo elemento s_n.

$\mathbf{d(s_n, s_1)}\ \theta\ \mathbf{r}$	$\mathbf{d(s_n, s_2)}\ \theta\ \mathbf{r}$	$\mathbf{d(s_n, s_1)}\ \theta\ \mathbf{d(s_n, s_2)}$	**Região**
$<$	$<$	$<$	I
$<$	$<$	$\geq$	II
$<$	$\geq$		III
$\geq$	$<$		IV
$\geq$	$\geq$	$<$	V
$\geq$	$\geq$	$\geq$	VI

Tabela 5.5: Condições determinantes para a definição da região de cobertura de um elemento s_n.

As condições definidas são utilizadas em algoritmos de inserção e também servem de base para os algoritmos de consulta. A próxima subseção define o processo de inserção de elementos na *MMH-tree*.

5.5.1 Construção

A *MMH-tree* foi projetada para operar em memória principal, então novos elementos são inseridos na estrutura sequencialmente. Para inserir um novo elemento s_n, o algoritmo navega na estrutura realizando uma consulta pontual, procurando pelo nó apropriedado para armazenar o novo elemento. Em cada nó, s_n é comparado com os pivôs (s_1, s_2) e a região adequada é determinada de acordo com a Tabela 5.5. Note-se que não há sobreposição entre as regiões de um mesmo nível da estrutura, então a escolha da região em cada nível da estrutura é feita de modo único e rápido.

Resumidamente, o algoritmo de inserção executa os passos seguintes. Partindo do nó raiz, se o nó visitado tem apenas um elemento, s_n é armazenado nesse nó e o algoritmo termina. Caso o nó tenha dois elementos, as distâncias de s_n aos pivots s_1 e s_2 são calculadas e a região adequada é determinada de acordo com as distâncias e o raio entre os pivôs r. Estas comparações seguem as condições descritas na Tabela 5.5. O processo é aplicado recusivamente em todos os níveis da estrutura.

O Algoritmo 9 descreve o procedimento de inserção em uma árvore MMH. Note que apenas dois cálculos de distância são necessários por nó para decidir em qual das seis subárvores o elemento s_n está coberto.

Algoritmo 9: Inserção na *MMH-tree*.

Insere (s_n, raiz N)

Entrada: s_n: elemento a ser inserido; N: raiz da subárvore

```
Sejam s1 e s2 os pivôs de N e r a distância d = (s1, s2).
se N não está cheio então
    Insira sn em N
senão
    d1 = d(s1, sn)
    d2 = d(s2, sn)
    se d1 < r então
        se d2 < r então
            se d1 < d2 então
                N = região I
            senão
                N = região II
            fim se
        senão
            N = região III
        fim se
    senão
        se d2 < r então
            N = região IV
        senão
            se d1 < d2 então
                N = região V
            senão
                N = região VI
            fim se
        fim se
    fim se
    se N não foi alocado então
        Aloque um novo nó
    fim se
    Insere(sn, N)
fim se
```

5.5.2 Consulta

Qualquer consulta por similaridade realizada pela *MMH-tree* depende da distância $d(s_q, \eta)$ para determinar se uma dada região deve ou não ser visitada. A Figura 5.10 mostra um exemplo de consulta por abrangência na *MMH-tree*. Para determinar se a bola de consulta $B_q(s_q; r_q)$ intersecta o hiperplano métrico η, é necessário calcular a distância $d(s_q, \eta)$, de acordo com a Equação 5.2. Neste caso, se $r_q \geq d(s_q, \eta)$, então a bola de consulta B_q intercepta a partição de s_2.

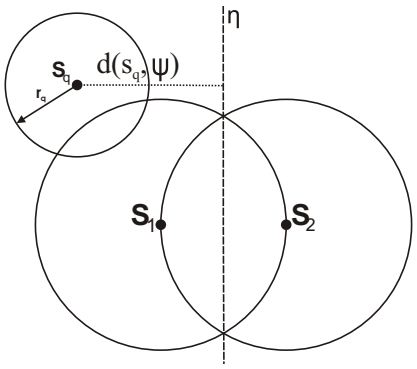


Figura 5.10: Um exemplo de consulta de agrangência na *MMH-tree*.

As consultas por abrangência e por vizinhos mais próximos podem ser respondidas pela *MMH-tree*. No caso da consulta aos vizinhos mais próximos, a ideia é definir um raio ativo de busca, que pode mudar conforme as regiões forem visitadas e vizinhos mais próximos forem atualizados. No caso de consultas por abrangência, este raio ativo é fixo e determinado pelo usuário no momento da consulta.

Uma vez que as consultas feitas na estrutura navegam em subárvores detectando sobreposições entre a bola de consulta $B(s_q, r_q)$ e as regiões do nó de cada nível consultado, é necessário definir estas condições de inteseção. Para descobrir qual região deve ser explorada em cada nó, o algoritmo de consulta deve testar as condições indicadas na Tabela 5.6, usando as seguintes subcondições

$$\mathbf{C}_1\colon\ (d(s_q, s_1) < r_q + r) \ \wedge\ (d(s_q, s_2) < r_q + r)$$

$$\mathbf{C}_2\colon\ (d(s_q, s_1) < r_q + r) \ \wedge\ (d(s_q, s_2) + r_q \geq r)$$

$$\mathbf{C}_3\colon\ (d(s_q, s_1) + r_q \geq r) \ \wedge\ (d(s_q, s_2) < r_q + r)$$

$$\mathbf{C}_4\colon\ (d(s_q, s_1) + r_q \geq r) \ \wedge\ (d(s_q, s_2) + r_q \geq r)$$

para determinar a intersecção da bola de consulta com as bolas. C_1 testa se a bola de consulta intersecta ambas as bolas $B_1(s_1, r)$ and $B_2(s_2, r)$; C_2 testa se a bola de consulta intersecta somente a bola $B_1(s_1, r)$; C_3 testa se a bola de consulta intersecta somente a bola $B_2(s_2, r)$ e C_4 testa se a bola de consulta não intersecta nenhuma bola do nó.

A Tabela 5.6 indica quais regiões as consultas por similaridade devem visitar enquanto navega-se na estrutura. Note que apenas dois cálculos de distância por nó são necessários para determinar as regiões adequadas para serem visitadas.

Região	Condição para sobreposição
I	$(C_1 \wedge (d_1 < d_2)) \vee (C_1 \wedge (d_1 \geq d_2) \wedge (r_q > d_\eta))$
II	$(C_1 \wedge (d_1 \geq d_2)) \vee (C_1 \wedge (d_1 < d_2) \wedge (r_q > d_\eta))$
III	C_2
IV	C_3
V	$(C_4 \wedge (d_1 < d_2)) \vee (C_4 \wedge (d_1 \geq d_2) \wedge (r_q > d_\eta))$
VI	$(C_4 \wedge (d_1 \geq d_2)) \vee (C_4 \wedge (d_1 < d_2) \wedge (r_q > d_\eta))$

Tabela 5.6: Condições para detectar sobreposição de bolas de consultas nas regiões da *MMH-tree*.

O Algoritmo 10 descreve a consulta por abrangência na *MMH-tree*. O algoritmo recebe um raio de consulta r_q e o centro de consulta s_q. Em cada nó, verifica-se se existem interseções da bola de consulta com as seis regiões definidas no nó (linhas 6 e 7), visitando as regiões cobertas (linha 8) recursivamente, baseado na distância de s_q aos pivôs (s_1, s_2). Em cada nó visitado, se a distância de s_q para um pivô for menor que r_q, então este pivô é adicionado à lista de resultados (linhas 4 e 5). Em cada nó, a interseção de regiões ocorre se alguma das condições descritas na Tabela 5.6 for atendida.

O Algoritmo 11 descreve a consulta aos k vizinhos mais próximos em uma *MMH-tree*. O algoritmo mantém uma lista de resultados de k elementos (sem considerar listas de empates) mais próximos a s_q, ordenados de acordo com sua a distância a s_q. Quando o algoritmo se inicia, a lista está vazia e ao raio ativo r_q é atribuído o valor infinito (ou valor do diâmetro do conjunto). O raio ativo é então reduzido cada vez que a lista é atualizada, ou seja, quando um elemento mais próximo é encontrado. O raio ativo assume o valor da distância de s_q ao k-ésimo elemento mais distante da lista. A lista final de resultados é formada pelos k elementos encontrados durante a navegação pela estrutura. Uma sobreposição ocorre se alguma das condições apresentadas na Tabela 5.6 for atendida, onde r_q é o raio ativo dinâmico

Algoritmo 10: Consulta por abrangência na *MMH-tree*

Range Query (s_q, r_q, $raiz$)

Entrada: s_q: centro de consulta; r_q: raio de consulta; $raiz$: ponteiro para a raiz.
Saída: $Lista$: lista de resultados.

```
1: se raiz == NULL então
2:     return
3: fim se
4: Calcule as distâncias d1 = d(s1, sq) e d2 = d(s2, sq) em raiz
5: se d1 ≤ rq então Lista.adicione(s1)
6: se d2 ≤ rq então Lista.adicione(s2)
7: se raiz não é um nó folha então
8:     Seja P a região onde sq pertence, de acordo com a Tabela 5.5.
9:     RangeQuery(sq, rq, P)
10:    para R = I..VI, R ≠ P faça
11:        se Bq(sq; rq) intersecta R então
12:            RangeQuery(sq, rq, R)
13:        fim se
14:    fim para
15: fim se
```

da consulta. Pelo Algoritmo 11, as linhas 16 a 18 testam as condições de interseção e visitam os nós adequados. Note que se s_q reside na região P, aquela região deve ser visitada em primeiro lugar, produzindo uma consulta em profundidade.

Técnica guiada

O raio ativo nas consultas aos k vizinhos mais próximos é reduzido conforme elementos mais próximos ao elemento de consulta s_q são encontrados. Quanto mais rápido este raio diminuir, mais subárvores são desconsideradas para análise pelo algoritmo, reduzindo o número de cálculos de distâncias necessários para responder a consulta.

Este processo pode ser otimizado na estrutura MMH e a técnica guiada para o algoritmo de $kNNq$ melhora o desempenho das consultas. Esse algoritmo escolhe uma sequência de visita em cada nó da *MMH-tree*, de forma a reduzir o raio ativo mais rapidamente. O algoritmo primeiro visita a região onde o elemento de consulta s_q pertence, e então visita as próximas regiões de acordo com distância deste às outras regiões. A sequência é formada pelas regiões mais próximas ao elemento de consulta.

A sequência apresentada na Tabela 5.7 leva a uma melhor configuração de visitas, e as condições são baseadas nos valores indicados pela Figura 5.11.

Algoritmo 11: Consulta aos k vizinhos mais próximos na *MMH-tree*.

Nearest Query (s_q, k, $raiz$)

Entrada: s_q: centro de consulta; k: número de vizinhos mais próximos; $raiz$: ponteiro para raiz.

Saída: $Lista$: lista de resultados.

```
1: se root == NULL então
2:     return
3: fim se
4: Calcule as distâncias d_1 = d(s_1, s_q) e d_2 = d(s_2, s_q) em raiz
5: se Lista.tamanho() < k então
6:     r_q = ∞
7: senão
8:     r_q = Lista[k].distãncia
9: fim se
10: para i = 1..2 faça
11:     se d_i < r_q então
12:         Lista.adicione(s_i)
13:         Lista.ordene()
14:         Lista.ElimineApos(k)
15:     fim se
16: fim para
17: se raiz não é um nó folha então
18:     Seja P a região onde s_q pertence, de acordo com a Tabela 5.5.
19:     NearestQuery(s_q, k, P)
20:     para R = I..VI, R ≠ P faça
21:         se B_q(s_q; r_q) intersecta R então
22:             NearestQuery(s_q, k, R)
23:         fim se
24:     fim para
25: fim se
```

Tabela 5.7: Sequência de regiões a serem visitadas no algoritmo de $kNNq$ guiado

		visiting order	
região em que s_q pertence	condição C	C é verdade	C é falso
I	$d_2 < 0.75r$	I→II→III→IV→(V,VI)	I→III→II→IV→(V,VI)
II	$d_1 < 0.75r$	II→I→IV→III→(V,VI)	II→IV→I→III→(V,VI)
III	$d_2 - r \leq r - d_1$	III→I→II→V→(IV,VI)	III→V→I→II→(IV,VI)
IV	$d_1 - r \leq r - d_2$	IV→II→I→VI→(III,V)	IV→VI→II→I→(III,V)
V	$d_\eta > 1.5r$	V→III→I→II→(IV,VI)	V→VI→III→I→(II,IV)
VI	$d_\eta > 1.5r$	VI→IV→II→I→(III,V)	VI→V→IV→II→(I,III)

5.6 Variações

Existem várias combinações possíveis entre as bolas centradas nos pivôs e o hiperplano métrico. A Figura 5.12 mostra algumas variações simétricas dentre as muitas que podem ser configuradas, levando a diferentes particionamentos. A nomenclatura das configurações define o número de regiões determinadas. Por exemplo, a estrutura $MMH_5 - tree$ divide

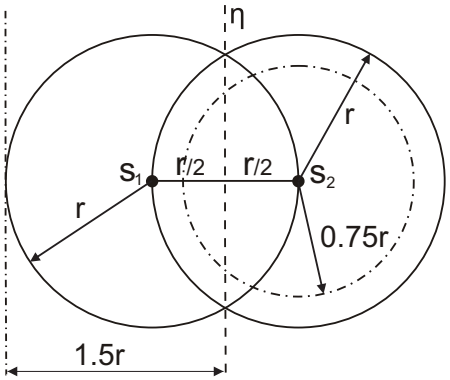


Figura 5.11: Medidas utilizadas para estimar a proximidade de elementos a regiões.

o espaço em cinco regiões. A alteração nos algoritmos de inserção e buscas para respeitar cada variação são pequenas, bastando considerar a fusão de regiões nas combinações.

Outra variação é modificar o raio da bolas centradas nos pivôs, gerando novas possibilidades para múltiplas bolas de cobertura que didivem o espaço de maneira disjunta. Essa abordagem é apresentada na seção seguinte.

5.6.1 A *Onion-tree*

Além das diferentes configurações geradas pelas combinações das bolas centradas nos pivôs e o hiperplano métrico, é possível modificar o raio das bolas centradas nos pivôs, gerando diferentes opções. A Figura 5.13 mostra duas opções de mudança de raio, metade e dobro do raio. O uso da metade do raio causou a diminuição do número de regiões no espaço, enquanto que o uso do dobro do raio em conjunto com o raio inicial aumentou o número de regiões formadas. A *Onion-tree* investiga o desempenho em consultas baseando-se na multiplicidade dos raios acima ou igual ao valor dois.

Deste modo, considerando-se a multiplicidade do raio a partir do dobro do raio, pode-se gerar várias camadas que aumentam o particionamento do espaço. A *Onion-tree* [Carélo et al., 2009] realiza essa divisão baseando-se na multiplicidade do raio dos pivôs iniciais da *MM-tree*, produzindo assim mais regiões de abrangência, levando a um maior particionamento do espaço métrico e portanto melhor desempenho em consultas por similaridade.

O número de expansões escolhido modifica o número de regiões em que o nó divide o

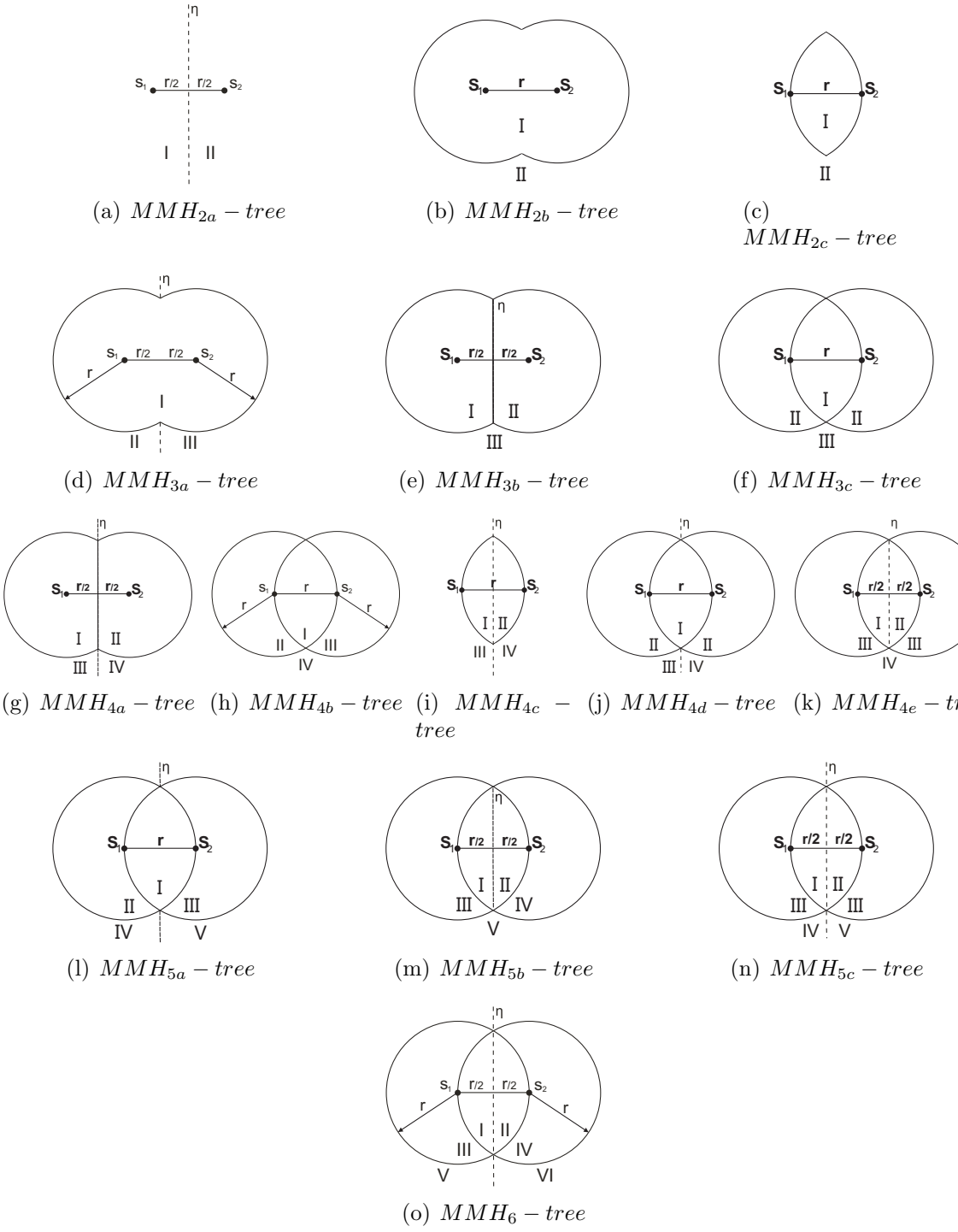
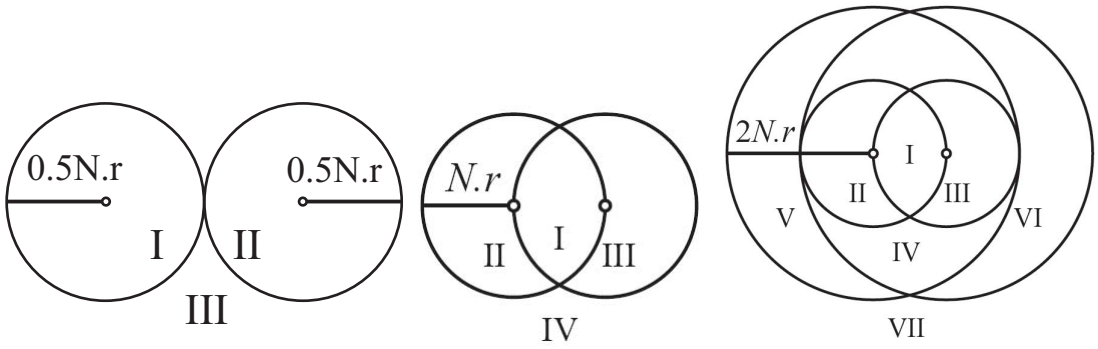


Figura 5.12: Diferentes combinações de regiões e o hiperplano métrico.

espaço e sua estrutura é diferente para cada número de expansões.

A Figura 5.14 mostra a estrutura do nó da *Onion-tree* de acordo com o número de expansões que o nó possui, baseando-se na multiplicidade do raio das bolas nos pivôs. Pela figura, cada expansão aumenta em três o número de regiões disjuntas. O número de regiões



(a) Metade do raio (b) Raio original (c) Dobro do raio

Figura 5.13: Variação do raio das bolas centradas nos pivôs.

geradas de acordo com o número de expansões é dado por

$$N\acute{u}mero\ de\ regiões = 3 * \ n\acute{u}mero\ de\ expansões\ +\ 4$$

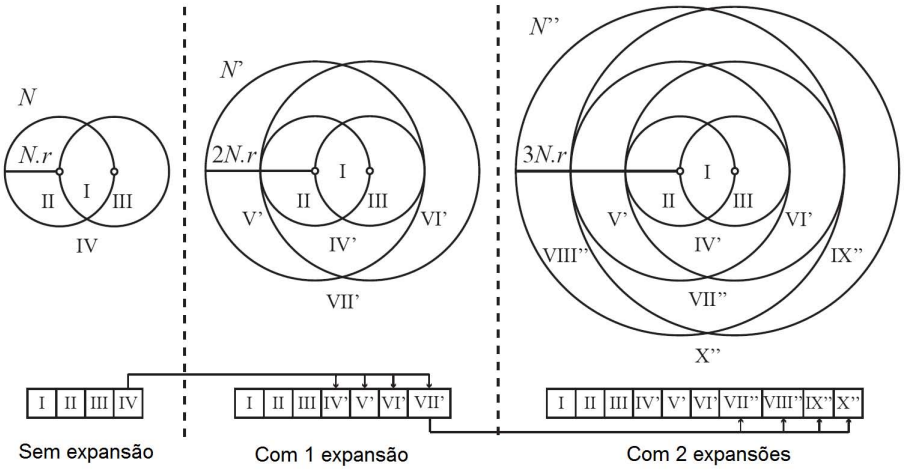


Figura 5.14: Comportamento do nó da *Onion-tree* de acordo com o número de expansões.

Existem duas abordagens para construir a *Onion-tree*, a fixa e a variável. A abordagem fixa (F-Onion) estabelece um número pré-determinado de expansões para todos os nós da estrutura, enquanto que a abordagem variável (V-Onion) calcula o número "ótimo" de expansões a cada nó criado na estrutura, baseando-se em critérios tais como a distância dos pivôs aos seus representativos. A *Onion-tree* também apresenta novas políticas para escolha dos pivôs e adaptações para a escolha da ordem de visita das novas regiões criadas. Maiores informações sobre as novas políticas e os algoritmos para esta variação podem ser encontrados no trabalho de Carélo [Carélo et al., 2009].

Foram realizados experimentos com diferentes formações de regiões utilizando-se diferentes expansões para cada conjunto de dados. A Figura 5.15 mostra os resultados de consultas $kNNq$ nos conjuntos de dados *Color Histograms* (histogramas coloridos de imagens) e *Brazilian Cities* (coordenadas latitude e longitude de cidades brasileiras). O número médio de cálculos de distância e o tempo gasto foi medido a partir de uma sequência de 500 consultas com centros escolhidos aleatoriamente dos conjuntos. Pode-se notar que a *Onion-tree* produziu melhor desempenho nas consultas, em suas duas políticas, fixa e variável, efeito causado pelo maior número de divisões baseando-se nos múltiplos raios de cobertura dos elementos. Em relação à abordagem fixa de raios, utilizaram-se cinco expansões para o conjunto *Brazilian Cities* e sete expansões para o conjunto *Color Histograms*.

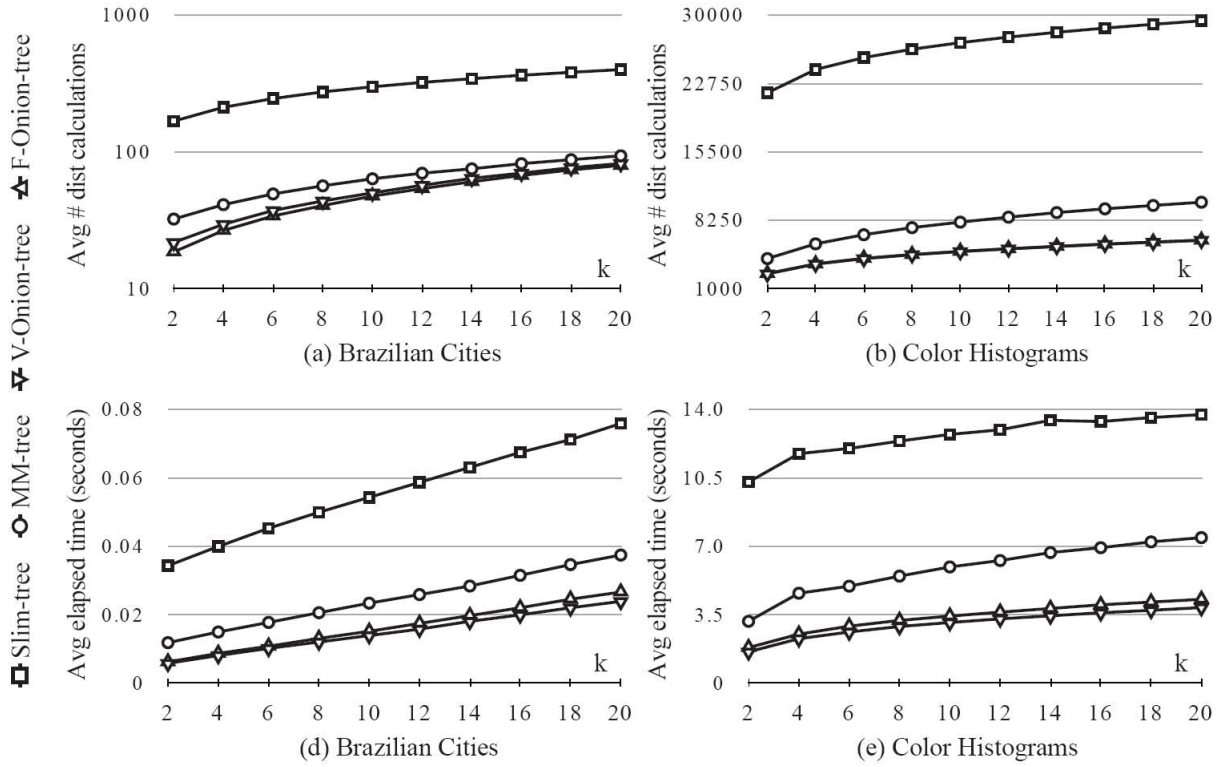


Figura 5.15: Número médio de cálculos de distância e tempo para execução de consultas $kNNq$ nas MAM *Onion-tree*, *MM-tree* e *Slim-tree*.

5.7 Experimentos

Esta seção mostra experimentos realizados comparando as estruturas aqui formuladas. Em específico, foram realizados experimentos envolvendo as estruturas *MB-tree*, *MMH-tree*, *MM-*

tree, *VP-tree* e *Slim-tree*. Embora a MAM *Slim-tree* tenha sido formulada para indexar dados em disco, aqui ela foi alterada para indexar dados em memória principal, de modo a manter os nós em memória.

Todas as estruturas comparadas aqui foram implementadas na linguagem C++ utilizando-se a biblioteca de estruturas métricas Arboretum [1]. Os experimentos foram realizados em um computador com processador Pentium D 3.4GHz com 2Gb de memória RAM. A *Slim-tree* foi construída utilizando-se as políticas *min-occupation* e MST (*minimum spanning tree*), considerada pelos autores as configurações padrão. O algoritmo Slim-down foi aplicado à *Slim-tree* em todos os conjuntos de dados utilizados. O tamanho da página utilizado na *Slim-tree* foi diferente para cada conjunto de dados, visando manter em média 50 elementos por nó. A *VP-tree* foi construída usando o algoritmo de *sampling* especificado pelo autor a uma taxa de 10% para a escolha dos pivôs (chamados de *vantage points*). Os conjuntos de dados utilizados não possuem ordem de formação dos elementos, sendo os elementos dispostos em ordem aleatória.

Quatro conjuntos de dados reais foram utilizados nos experimentos, conforme detalhados na Tabela 5.8. Os experimentos visam analisar o desempenho das estruturas em duas situações: construção e consultas. Árvores em memória devem ser rápidas para se construir pois elas podem ser utilizadas para agilizar subconsultas de uma base de dados. O tempo de consulta deve ser pequeno também, pois elas têm maior flexibilidade de particionar o espaço sem a exigência de manter vários elementos por nó.

A primeira análise, realizada para mensurar o desempenho da construção das estruturas, mede o número de cálculos de distância e o tempo total gasto para construir cada estrutura para cada conjunto de dados. Assim, quanto mais rápida a estrutura é construída, melhor, pois já a habilita para consultas.

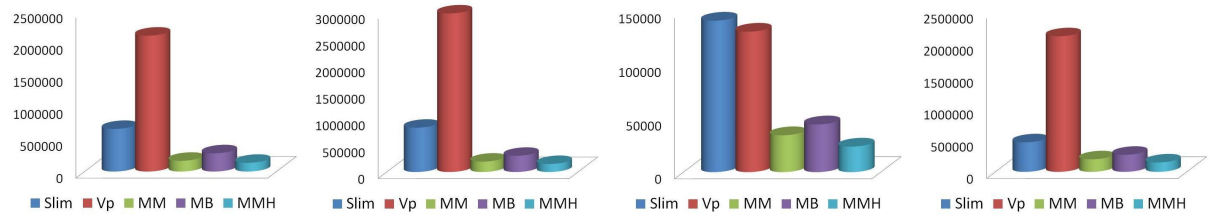
As Figuras 5.16 e 5.17 mostram gráficos comparativos do desempenho de construção de cada MAM. A Figura 5.16 mostra o número de cálculos de distância necessários para construir cada MAM, enquanto a Figura 5.17 realiza a mesma comparação porém mostrando o tempo total gasto.

Note-se que uma característica importante entre as estruturas comparadas nesta estapa

[1] http://gbdi.icmc.usp.br/arboretum

Nome	Tamanho	# Dimensões	Métrica	Descrição
ColorHisto	10000	32	L_2	Histograma de cores extraído de uma coleção de imagens (http://kdd.ics.uci.edu).
EigenFaces	11900	16	L_2	vetores descritivos de faces do Projeto Informedia [Wactlar et al., 1996] da Universidade Carnegie Mellon.
Currency	2311	6	L_2	Taxas de câmbio diárias para as moedas USD, HKD, BP, FRF, DEM, JPY
MetricHisto	10000	-	MHD	Histogramas métricos de imagens médicas em níveis de cinza [Traina et al., 2003b].

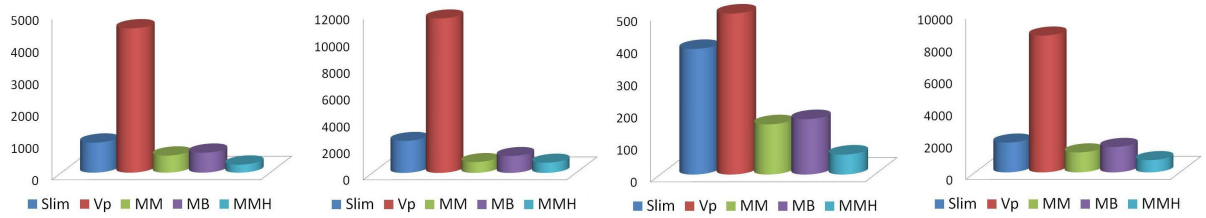
Tabela 5.8: Conjuntos de dados usados nos experimentos



(a) ColorHisto (b) EigenFaces (c) Currency (d) MetricHisto

Figura 5.16: Número de cálculos de distância para construção dos MAM sobre os conjuntos de dados.

de experimentos é que a *VP-tree* é a única estrutura estática, onde todo o conjunto de dados é considerado e analisado para montar a estrutura, não permitindo inserções posteriores. Esta estratégia provê à *VP-tree* bom desempenho nas consultas, mas a sua construção teve o pior desempenho frente as outras estruturas comparadas, de acordo com a Figura 5.16. Isso se dá pelo algoritmo de *sampling* aplicado na determinação dos melhores pivôs em cada nó durante o processo de construção. A estrutura que teve melhor desempenho nesta etapa foi a *MMH-tree*, considerando todos os conjuntos de dados. Um resultado peculiar ocorre no gráfico Figure 5.16(c), onde a *Slim-tree* realiza mais cálculos de distância que a *VP-tree*. Uma investigação foi realizada neste teste, e verificou-se que o algoritmo Slim-Down não produziu melhoramentos na estrutura, representando cálculos de distâncias adicionais na construção do conjunto Currency.



(a) ColorHisto (b) EigenFaces (c) Currency (d) MetricHisto

Figura 5.17: Tempo gasto (ms) para construção das estruturas sobre todos conjuntos.

Pelos gráficos mostrados na Figura 5.17, podemos observar o tempo total que cada MAM gasta para indexar cada conjunto de dados. Quase todos os gráficos preservam o mesmo comportamento dos gráficos do número de cálculos de distância, pois no caso de estruturas em memória, o impacto no tempo é diretamente afetado pelo número de cálculos de distância realizado. Já nas estruturas projetadas para disco, o número de acessos a disco para recuperar os nós afetaria o tempo de construção.

O gráfico mostrado na Figura 5.17(c) apresentou um comportamento diferente ao considerar a *Slim-tree* e a *VP-tree*, em relação ao gráfico da Figura 5.16(c), onde a *VP-tree* gastou mais tempo para construir a estrutura que a *Slim-tree*, porém realizou menos cálculos de distâncias. Este comportamento no tempo é explicado pelo cálculo custoso envolvido no algoritmo de inserção da *VP-tree*, que envolve cálculos estatísticos como a mediana para escolher o melhor pivô de cada nó construído, constribuindo no aumento de tempo na construção.

O segundo experimento compara o desempenho de cada MAM para realizar consultas por abrangência e aos k vizinhos mais próximos em todos os conjuntos de dados. Novamente, foi medido o número de cálculos de distância e o tempo gasto para as consultas. Os gráficos que medem o número de cálculos de distância representam a média obtida ao realizar 500 consultas com diferentes centros de consulta para cada raio ou k. Foi considerado que todos os elementos de consulta são indexados nas estruturas, e foram escolhidos de maneira aleatória, sendo os mesmos centros para todos tipos de consultas e estruturas. Os gráficos de tempo para as consultas mostram o tempo total gasto para responder a estas 500 consultas.

As Figuras 5.18 e 5.19 mostram os resultados obtidos nesta etapa. O raio máximo das consultas por abrangência, apresentados em escala logarítmica nos gráficos, representa o raio que recupera aproximadamente 10% dos elementos similares obtidos pela consulta.

Os gráficos das Figuras 5.18 e 5.19 mostram que a *MMH-tree* antigiu o melhor desempe-

nho em todos os testes. Isto se dá pelo fato do melhor particionamento do espaço conseguido com a inclusão do hiperplano métrico, criando mais regiões disjuntas que a *MM-tree*. Um resultado muito interessante é o desempenho da árvore binária métrica, a *MB-tree*, que alcançou melhor desempenho que a *Slim-tree* em todos os testes, tendo seu melhor desempenho indexando o conjunto de dados Currency.

5.8 Conclusão

Aplicações emergentes necessitam que o SGBD disponha de mecanismos de indexação de dados que possam ser mantidos em memória principal. O aumento da capacidade da memória principal e seu custo cada mais reduzido alavancaram o desenvolvimento de estruturas de indexação em memória principal, levando a estruturas mais simples e mais rápidas em construção e consultas. Enquanto as estruturas de dados destinadas para memória secundária impõem que a estrutura do nó da árvore mantenha vários elementos por nó, as estruturas em memória principal não possuem essa imposição, podendo particionar o espaço com menos elementos mais livremente. Elas respondem mais rápido a consultas por similaridade, também por não realizam acessos a disco para tal.

Neste capítulo foi mostrado o uso do mapeamento de espaços métricos em espaços multidimensionais como ferramenta para dispor de mais propriedades que são utilizadas para melhorar o particionamento do espaço de dados. Pela aplicação da lei dos cossenos, foi mostrado como medir distâncias de elementos a hiperplanos, tornando-os hiperplanos métricos. A combinação do hiperplano métrico e as bolas definidas pelos pivôs da *MM-tree* levaram a um melhor particionamento do espaço métrico, levando a desempenhos melhores das estruturas. Além de descrever a *MM-tree*, as estruturas de árvore binária métrica (hiperplano métrico como uma MAM) e a *MMH-tree* (hiperplano métrico aplicado à *MM-tree*) foram descritas como aplicações das propriedades do espaço mapeado.

Experimentos utilizando quatro diferentes conjuntos de dados reais foram realizados para estudar e analisar os processos de construção e consulta das estruturas propostas. Os resultados dos experimentos de construção mostraram que a *MMH-tree* teve o melhor desempenho, realizando até 95% menos cálculos de distância quando comparada com a *VP-tree* (a mais

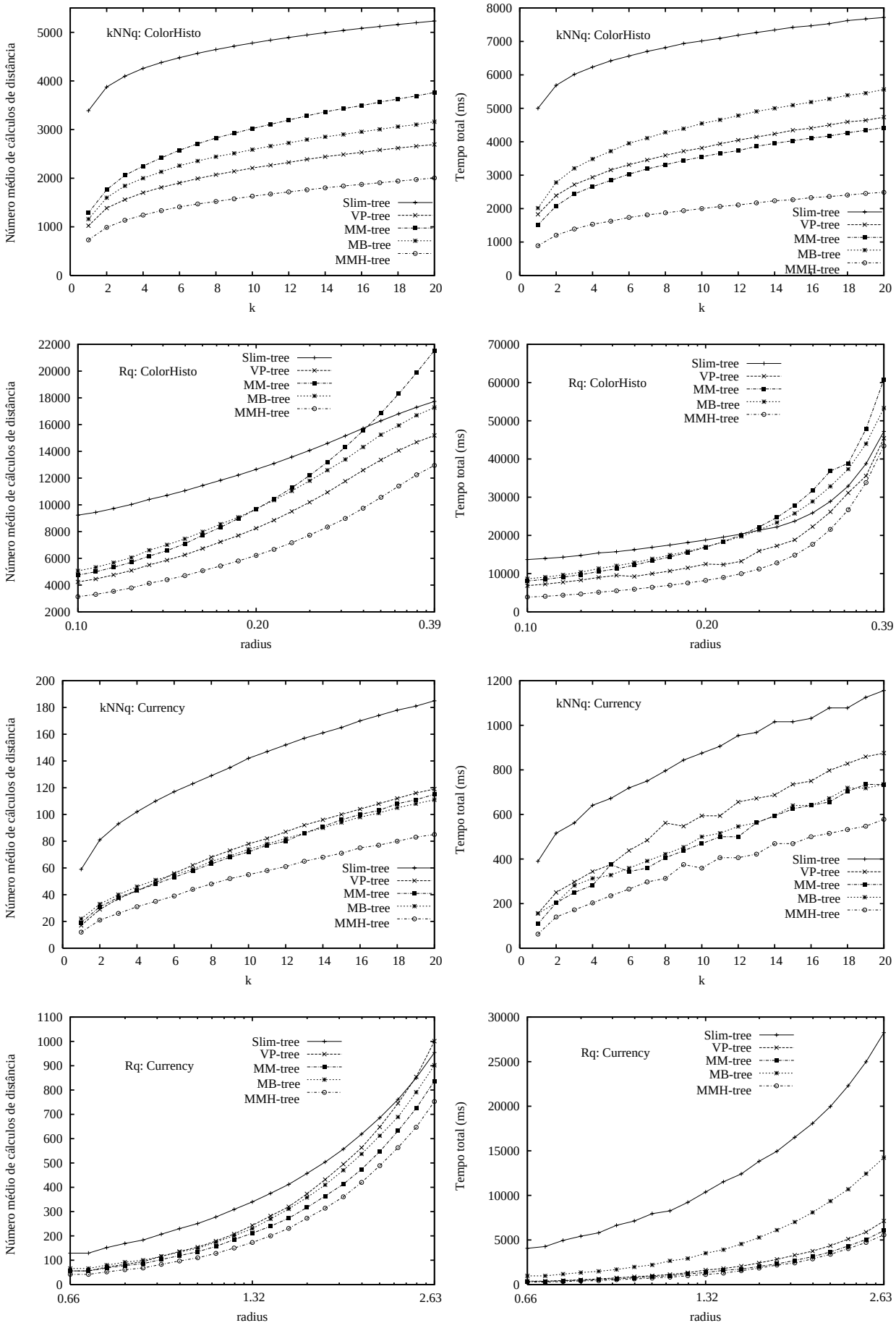


Figura 5.18: Desempenho de consultas por abrangência e consulta aos k vizinhos mais próximos nos conjuntos ColorHisto e Currency.

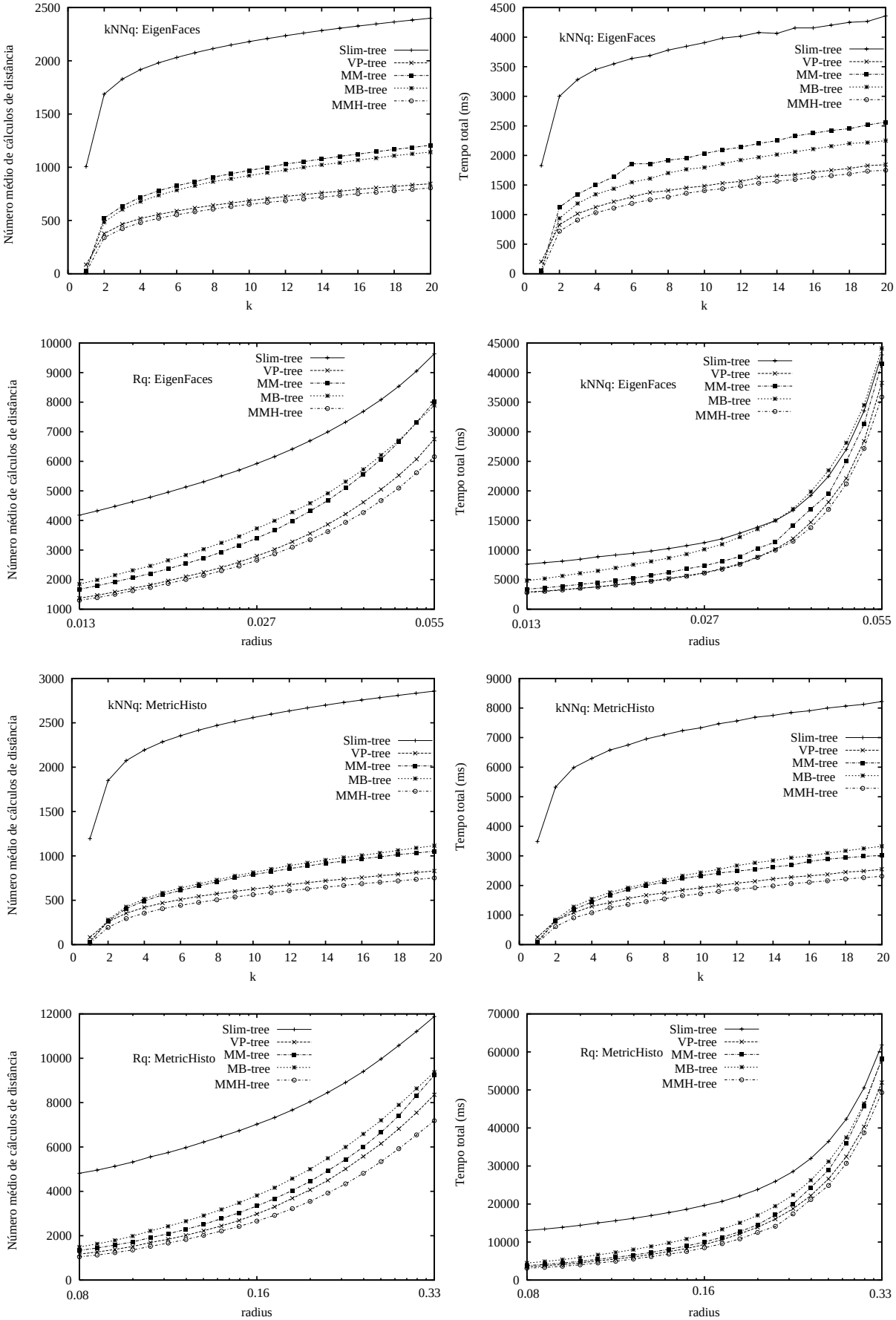


Figura 5.19: Desempenho de consultas por abrangência e consulta aos k vizinhos mais próximos nos conjuntos EigenFaces e MetricHisto.

lenta do teste em construção) com ganho em tempo de 93%. Quando comparada à *MM-tree* (a segundo mais rápido em construção nos testes), a *MMH-tree* realizou até 31% menos cálculos de distância, com ganho em tempo de 64%.

Os resultados do desempenho das consultas mostraram que a *MMH-tree* também teve o melhor desempenho frente as outras estruturas, realizando até 26% menos cálculos de distância quando comparada com a *VP-tree*, com ganho de 48% em tempo, e até 47% menos cálculos de distância quando comparada à *MM-tree*, com ganho de tempo de 44%.

Capítulo

6

Técnicas de deformação de espaços métricos

6.1 Introdução

A maioria das técnicas de recuperação de dados complexos por conteúdo envolve a extração das características e a escolha da função de distância mais adequada a este extrator. Assim, um conjunto de elementos $S \subset \mathbb{S}$ é formado e associado a uma função de distância formando um espaço métrico.

A eficiência de um MAM está em como ele particiona o espaço, como visto no Capítulo 5, e a qualidade das consultas depende da função de distância escolhida [Bugatti et al., 2008]. Quanto mais características significativas são extraídas dos dados complexos, maior é a habilidade de identificar os mais similares a um dado elemento de consulta. Entretanto, quanto mais características são adicionadas ao vetor de características que representa o elemento, maior será a dimensionalidade deste, levando à chamada "maldição da dimensionalidade". Esta condição afeta o resultado de consultas e degrada as estruturas de acesso [Beyer et al., 1999a].

Os trabalhos que visam reduzir a dimensionalidade dos dados não são úteis para indexação, pois eles causam distorções no espaço das distâncias e nas respostas, que não são facilmente previsíveis, impossibilitando o uso de técnicas de correção [Aggarwal et al., 2001,

Korn et al., 2001]. Ao invés disso, o uso de mapeamentos com funções de mapeamento estritamente não-decrescentes pode ser usado e leva à uma explosão do espaço resultante, onde elementos mais próximos preservam a proximidade, e elementos gradativamente mais distantes vão ficando ainda mais distantes no espaço mapeado. Neste capítulo se investiga esse efeito, visando auxiliar a contrabalançar a maldição da alta dimensionalidade, para evitar que muitos elementos tenham distâncias com valores semelhantes, evitando ou atrasando a ocorrência da saturação das estruturas internas dos algoritmos de busca. De fato, um estudo empírico recentemente publicado deu indícios de que isso de fato pode ocorrer [Nadvorny & Heuser, 2004].

Este capítulo mostra uma técnica para melhorar o desempenho das MAMs quando indexam conjuntos de alta dimensionalidade ou que possuam os efeitos dela. A técnica consiste em deformar o espaço métrico para atenuar os efeitos da maldição da dimensionalidade. A técnica aplica uma função monotonicamente crescente como uma "função de deformação", que muda o espaço métrico, produzindo um novo espaço de distâncias. Este novo espaço de distâncias não é um espaço métrico, mas ele possui propriedades que são derivadas de um espaço métrico. Tais propriedades podem ser utilizadas nas consultas para acelerar o processo de busca, substituindo as propriedades de uma métrica por outras duais, que levam a um melhor desempenho.

6.2 Deformando o espaço métrico

Ao invés de modificar as estruturas para tentar melhorar os resultados obtidos pelos MAMs, foi realizada uma mudança no espaço das distâncias. Note que qualquer alteração no espaço das distâncias deve preservar a ordem das distâncias dos elementos, para que não haja perturbação nos resultados. Enquanto no espaço das distâncias se pode realizar estas operações, no espaço das características isso não pode ser realizado, pois distorce a distribuição real dos dados. Uma simples normalização aplicada diretamente no conjunto das características modifica totalmente o resultado das consultas quando comparado ao espaço original.

A abordagem da mudança no espaço das distâncias aqui aplicada tem como objetivo inibir os efeitos da alta dimensionalidade usando uma técnica que deforma o espaço métrico

por itermédio de uma "função de deformação", de modo que pequenas distâncias são comprimidas e grandes distâncias são expandidas.

A função de deformação é uma função estritamente crescente $f : Im(d) \subseteq \mathbb{R}^+ \rightarrow \mathbb{R}^+$, onde d é a distância. Uma função entre dois conjuntos ordenados é monótona quando ela preserva (ou inverte) a relação de ordem. Quando a função preserva a relação, ela é chamada de função crescente e quando ela inverte a relação ela é chamada de função decrescente. Usa-se o prefixo "estritamente" para enfatizar que a função é injetiva, ou seja, não existe a derivada nula no corpo da função. Para funções estritamente crescentes, como na Figura 6.1, $b > a \rightarrow f(b) > f(a)$.

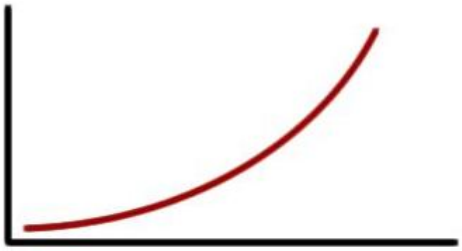


Figura 6.1: Exemplo de função estritamente crescente.

Para ilustrar o resultado da aplicação da função de deformação, considere a Figura 6.2. A figura mostra a função de deformação f_s usada para comparar os elementos (s_c, s_1, s_2, s_3), que são ordenados de acordo com suas distâncias ao elemento s_c através da função de distância d, que é chamada de distância original. Desta maneira, a função f_s comprime/expande a distância original em um espaço de distâncias deformado.

Para enfatizar o efeito, distâncias produzidas por $f_s \circ d$ são mostradas na Figura 6.2 normalizadas pela maior distância dentre os elementos. Como pode ser notado, os elementos que estão próximos de s_c se tornam ainda mais próximos e os elementos que estão distantes de s_c se tornam ainda mais distantes. Desta maneira, a função de deformação reduz o efeito da maldição da alta dimensionalidade em espaços de alta dimensão.

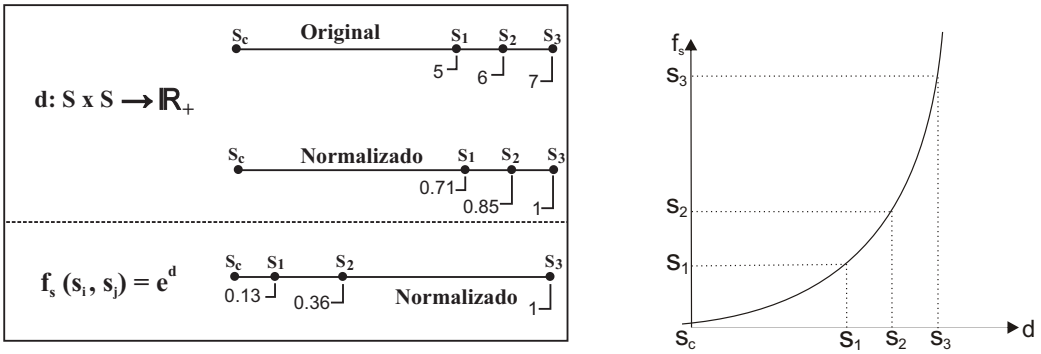


Figura 6.2: O efeito de deformar do espaço das distâncias.

A aplicação da função de deformação é equivalente a um mapeamento de espaços: a função de deformação mapeia o espaço original de distâncias centrado num dado elemento em outro espaço, produzindo o espaço deformado das distâncias. Diferentemente das técnicas de redução de dimensionalidade, o espaço deformado preserva todos atributos dos vetores de características e todos contribuem igualmente à distribuição dos dados.

Deve ser notado que o espaço deformado das distâncias não é métrico, apesar de preservar a ordem das distâncias do espaço de distâncias original. Entretanto, as diferenças das distâncias entre pares de elementos que se movimentarem no espaço das distâncias são governadas pela função de deformação. Assim, é possível calcular as diferenças das distâncias como um erro quando se calculam as distâncias. Um mapeamento que garanta um limite de erro máximo (α) é um conceito algébrico bem conhecido da teoria dos espaços métricos, chamado mapeamento contínuo α-Lipschitz.

Aqui, restringe-se a análise da função de deformação que precisa ser uma função estritamente crescente, para a função $f_s(d) = b^d$, onde $b > 1$ é a base da deformação. Deste modo, o mapeamento Lipschitz definido por valores reais ($\mathbb{R}$) das distâncias é limitado por uma constante α definida como

$$\alpha \geq \frac{|b^{d_1} - b^{d_2}|}{|d_1 - d_2|}$$

para todos $d_1, d_2 \in \mathbb{R}^+$ e $b > 1$.

Para exemplificar como a distribuição das distâncias muda pela função de deformação, a Figura 6.3 mostra o mesmo experimento da Figura 3.7 agora usando a função de deformação $f_s = e^d$ (usando o número de Euler e como a base de deformação). Como pode ser visto, agora a diferença das distâncias máximas e mínimas em um conjunto de 128 dimensões muda de 1/2 para 1/7, isto é, similar à proporção da distribuição do espaço original no conjunto de 16 dimensões, onde o efeito da maldição da dimensionalidade é bem mais baixo.

As propriedades do espaço mapeado podem ser derivadas do espaço original. A próxima seção apresenta as propriedades deste novo espaço de distâncias, relevantes para a criação e consulta das estruturas, detalhando o processo da derivação das propriedades, e apresentando a formulação geral para qualquer espaço deformado por uma função exponencial.

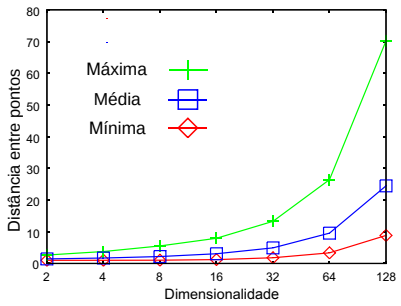


Figura 6.3: Distâncias mínimas, máximas e médias deformadas pela função $f_s = e^d$.

6.2.1 Propriedades do espaço deformado

Nesta seção é feita a análise das propriedades derivadas de um mapeamento contínuo α-Lipschitz resultante da aplicação de uma função exponencial a uma função de distância d (que satisfaz os axiomas de identidade, simetria, não negatividade e desigualdade triangular), para descobrir propriedades que possam ser úteis para os algoritmos de MAM para realizar podas de subárvores.

Considerando-se uma função de deformação definida como $f_s(s_i, s_j) = b^{d(s_i,s_j)}$, onde $b \in \mathbb{R}^+, b > 1$ e d é uma métrica qualquer. O mapeamento do espaço das distâncias produz um novo espaço que satisfaz as propriedades que seguem.

Identidade - Em espaços métricos, a propriedade de identidade dita que

$$d(s_i, s_j) = 0 \Rightarrow s_i = s_j$$

No espaço deformado das distâncias, a propriedade de identidade é modificada para a seguinte expressão

$$f_s(s_i, s_j) = 1 \Rightarrow s_i = s_j$$

porque $b^0 = 1$

Propriedades de simetria e não-negatividade - A propriedade de simetria original é válida para espaços deformados também, uma vez que

$$f_s(s_i, s_j) = b^{d(s_i,s_j)} = b^{d(s_j,s_i)} = f_s(s_j, s_i)$$

assim como a propriedade de não-negatividade:

$$0 < f_s(s_i, s_j) = b^{d(s_i,s_j)} < \infty$$

Propriedade de limitação - A propriedade de desigualdade triangular não é válida no espaço deformado de distâncias. Mas dela pode ser derivada a uma propriedade útil para MAM, permitindo aos algoritmos utilizá-la em suas operações internas. Utilizamos a propriedade de desigualdade triangular do espaço original para encontrar os limites inferiores e superiores que o espaço deformado das distâncias deve satisfazer para pares de três elementos. A partir da propriedade de desigualdade triangular, podemos escrever

$$\underbrace{|d(s_1, s_3) - d(s_3, s_2)|}_{limite\ inferior} \leq d(s_1, s_2) \leq \underbrace{d(s_1, s_3) + d(s_3, s_2)}_{limite\ superior} \tag{6.1}$$

para quaisquer $s_1, s_2, s_3 \in D$. Aplicando a função de deformação, podemos derivar a propriedade de limitação gerada pela função de deformação $f_s(s_1, s_2) = b^{d(s_1,s_2)}$, o que resulta em

$$b^{|d(s_1,s_3)-d(s_3,s_2)|} \leq b^{d(s_1,s_2)} \leq b^{d(s_1,s_3)+d(s_3,s_2)} \ .$$

Definindo $f_s^{-1}(s_i, s_j) = \log_b(f_s(s_1, s_2)) = d(s_i, s_j)$, temos

$$\underbrace{b^{|f_s^{-1}(s_1,s_3)-f_s^{-1}(s_3,s_2)|}}_{limite\ inferior} \leq f_s(s_1, s_2) \leq \underbrace{b^{(f_s^{-1}(s_1,s_3)+f_s^{-1}(s_3,s_2))}}_{limite\ superior}. \tag{6.2}$$

Esta propriedade permite estabelecer os limites inferiores e superiores para distâncias numa dada triangulação no espaço deformado das distâncias. Esta propriedade pode ser aplicada para o descarte de subárvores não necessárias para a visita dos nós (poda de subárvores).

Embora possa se escolher qualquer tipo de função exponencial de deformação, uma formulação particular pode ser escrita para cada função de deformação. Por exemplo, considere-se uma função de deformação da forma $f_s(s_i, s_j) = e^{d(s_i,s_j)}$. Então, as seguintes propriedades valem no espaço deformado gerado.

1. Identidade: $f_s(s_i, s_j) = 1 \Rightarrow s_i = s_j$;

2. Simetria: $f_s(s_i, s_j) = f_s(s_j, s_i)$;

3. Não-negatividade: $0 < f_s(s_i, s_j) < \infty$;

4. Limitação: $e^{|ln(f_s(s_1,s_3))-ln(f_s(s_3,s_2))|} \leq f_s(s_1, s_2) \leq e^{(ln(f_s(s_1,s_3))+ln(f_s(s_3,s_2)))}$

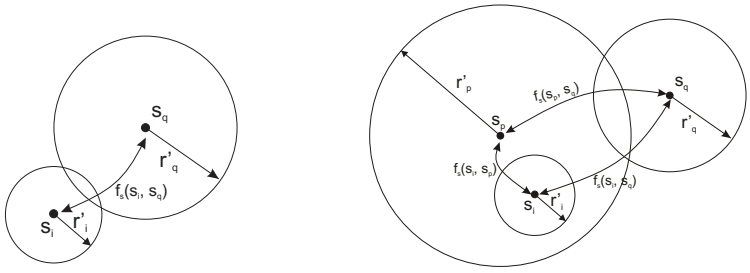
6.2.2 Consultas nos espaços mapeados

As consultas por similaridade em espaços deformados de distâncias podem se aproveitar da propriedade de limitação para podar subárvores durante a execução da consulta. Para aplicar a técnica proposta em um MAM, este deve ser adaptado de modo que as novas propriedades do espaço mapeado sejam aplicadas no lugar das pertinentes ao espaço original. Embora as mudanças sejam diretamente aplicáveis na implementação, para completude do método o algoritmo de consulta por abrangência (Rq) adaptado para utilizar as novas propriedades do espaço deformado das distâncias para podar subárvores é mostrado nesta seção. Chamamos esta nova estrutura de *Slim-tree XS* (do inglês *e**X**panded **S**pace*).

As distâncias $d(s_p, s_i)$ de cada elemento s_i ao representante s_p são armazenadas nós respectivos nós, seguindo a estrutura da *Slim-tree*, para reduzir o tempo gasto para recalcular distâncias. Assim, nas fórmulas seguintes, marcamos as distâncias já armazenadas como $\breve{d}$. Como o cálculo do mapeamento das distâncias também é uma operação custosa, propõe-se aqui modificar a estrutura do índice para armazenar também o valor deformado das distâncias, de modo pré-calculado durante a construção do índice, demarcado como $\breve{f}_s$. Embora esta modificação reduza a capacidade do nó, será mostrado nos experimentos que isso é compensado pelo melhor desempenho da estrutura.

Para efeito de ilustração, a Figura 6.4(b) exemplifica uma consulta em um espaço euclidiano de duas dimensões. Utilizamos a notação (s_i, r'_i) para representar uma bola em um espaço deformado de distâncias centrada no elemento s_i com raio de cobertura r'_i, onde $r'_i = b^{r_i}$. Para determinar se uma bola de consulta (s_q, r'_q) intersecta os elementos armazenados em um nó, de modo a intersectar a bola (s_i, r'_i), é necessário verificar se $f_s(s_i, s_q) \leq r'_i + r'_q$. Se esta condição for verdadeira, as bolas se intersectam e a região (subárvore) da bola (s_i, r'_i) deve ser visitada, como a Figura 6.4(a) indica. Entretanto, podemos evitar o cálculo de $f_s(s_i, s_q)$ usando a nova propriedade de limite inferior e as distâncias armazenadas. A verificação

$b^{|\check{f}_s{}^{-1}(s_i,s_p)-f_s^{-1}(s_p,s_q)|} > r'_i + r'_q$ permite identificar que a bola de consulta não intersecta a bola (s_i, r'_i). Se a condição não for verdadeira, a distância $f_s(s_i, s_q)$ deve ser calculada para determinar se a interseção ocorre.



(a) (b)

Figura 6.4: (a) Intersecção de bolas em um espaço deformado de distâncias. (b) Exemplo de uma consulta $Rq(s_q, r'_q)$.

O algoritmo de consulta por abrangência para espaços deformados de distâncias é apresentado no Algoritmo 12. A consulta por abrangência $Rq(s_q,\, r'_q)$ seleciona todos elementos s_j tais que $f_s(s_j, s_q) < r'_q$. O algoritmo inicia no nó raiz e visita toda subárvore que não pôde ser excluída pela propriedade de limitância, como indicado na Seção 6.2.1.

Algoritmo 12: Consulta por abrangência na *Slim-tree XS*.

Range Query (s_q, r'_q, *raiz*)

Entrada: s_q: centro de consulta; r'_q: raio de consulta; *raiz*: raiz da subárvore.

```
1:  Seja s_p o elemento pai de raiz
2:  se raiz não é nó folha então
3:      para todo s_j em raiz faça
4:          se b^{|f̌_s^{-1}(s_i,s_p) − f_s^{-1}(s_p,s_q)|} ≤ r'_i + r'_q então
5:              Calcule f_s(s_j, s_q);
6:              se f_s(s_j, s_q) ≤ r'_i + r'_q então
7:                  Range Query(s_q, r'_q, raiz.subárvore(s_j));
8:              fim se
9:          fim se
10:     fim para
11: senão
12:     para todo s_j em raiz faça
13:         se b^{|f̌_s^{-1}(s_i,s_p) − f_s^{-1}(s_p,s_q)|} ≤ r'_q então
14:             Calcule f_s(s_j, s_q);
15:             se f_s(s_j, s_q) ≤ r'_q então
16:                 Adicione oid(s_j) ao resultado;
17:             fim se
18:         fim se
19:     fim para
20: fim se
```

Uma consideração final quanto aos detalhes de implementação desta técnica é quanto à normalização do espaço das distâncias, mostrada na Figura 6.2. Teoricamente, a norma-

lização não é necessária e foi utilizada no exemplo para explicar o método. Mas vale lembrar que uma função exponencial pode gerar valores muito grandes se o espaço original também os possui. Assim, pode ser considerado, para estes casos, realizar uma amostragem dos elementos do espaço original e utilizar este conjunto reduzido para normalizar as distâncias antes de aplicar a função de deformação, evitando-se assim um estouro numérico no cálculo da exponencial. Note-se que uma aproximação para um hipercubo unitário das distâncias é suficiente pois previne estouro numérico e não produz alteração nos resultados.

6.3 Experimentos

Esta seção apresenta os experimentos realizados para analisar o efeito da deformação do espaço das distâncias em MAMs construídos a partir de conjuntos de dados tanto sintéticos como reais. Foi utilizada a MAM *Slim-tree* para medir o número de acessos a disco, o número de cálculos de distâncias efetuados e o tempo total para se realizar consultas por similaridade em ambos os espaços, no espaço métrico original e no mapeado. A *Slim-tree* utilizada nos experimentos, bem como sua adaptação para adequar o mapeamento do espaço, está disponível na biblioteca Arboretum [1], escrita em C++. Os testes foram realizados em um computador com um processador Pentium D 3.4Ghz e 2Gb de memória RAM. A *Slim-tree* foi construída utilizando-se as políticas de *min-occupation* e MST, consideradas padrão pelos autores da mesma. Os conjuntos de dados utilizados não possuem ordem de formação dos elementos, sendo os elementos dispostos em ordem aleatória.

Duas estruturas foram construídas para cada conjunto de dados: uma baseada no espaço métrico original (referenciada aqui de **Slim-tree**), e outra baseada na estrutura modificada para indexar os dados num espaço de distâncias mapeado (referenciada aqui como **Slim-tree XS**). Nos experimentos, o tamanho da página utilizado nas estruturas é diferente para cada conjunto de dados, de modo a manter nós ocupados em torno de 50 elementos. Para um determinado conjunto, as estruturas possuem o mesmo tamanho de página, possivelmente levando a uma ocupação menor da *Slim-tree XS* em relação à *Slim-tree*, pelo fato que a primeira armazena também distâncias no espaço deformado. Todos os experimentos foram

[1] http://gbdi.icmc.usp.br/arboretum

realizados utilizando um conjunto de 500 consultas com diferentes centros de consulta. Para cada experimento foi medido o número médio de acessos a disco, o número médio de cálculos de distância e o tempo total necessário para realizar 500 consultas. Os elementos de consulta são randomicamente escolhidos dentre os indexados na estrutura.

Três tipos de conjuntos foram usados nos experimentos, sendo um sintético e dois reais, como descrito a seguir.

***Synthetic*:** Sete conjuntos sintéticos, cada um tendo 10.000 pontos, gerados em um hipercubo unitário sob uma distribuição uniforme, com 2, 4, 8, 16, 32, 64 e 128 atributos (dimensões). Para estes conjuntos foi utilizada a distância euclidiana. Cada conjunto leva ao efeito mais forte da maldição da alta dimensionalidade naquela dimensionalidade, pois em um conjunto criado como estes não existem correlações entre os atributos.

***EnglishWords*:** Uma amostragem aleatória com 24.893 palavras da língua inglesa, sob a função de distância L_{edit}. Este caso também é um desafio grande pois a função de distância retorna valores discretos, levando a um número grande de empates. O efeito é o mesmo da maldição da alta dimensionalidade pois como as distâncias são discretas e variam pouco, qualquer aumento no raio de consulta engloba muitos elementos (nós) na estrutura.

***PCA*:** Um conjunto de 17.162 imagens projetadas em um espaço ortonormal de 43 autovetores (PCA) definidos a partir de um conjunto de treinamento formado por amostragem linear. Foi utilizada a distância euclidiana para este conjunto. Este conjunto mostra um comportamento médio dos conjuntos reais baseado nos vetores de características, onde seus atributos apresentam algum grau de correlação.

6.3.1 Avaliando o efeito da dimensionalidade

O primeiro experimento compara diferentes funções de deformação com o espaço original de distâncias utilizando os conjuntos sintéticos (*Synthetic*). Estes conjuntos foram escolhidos para explorar a dimensão intrínseca alta induzida pela distribuição uniforme, pois

como não há correlação entre os atributos, a maldição da alta dimensionalidade será grande, prejudicando ao extremo o desempenho das estruturas de indexação.

As seguintes funções potência foram testadas

Power2: $f_s(s_i, s_j) = 2^{d(s_i,s_j)}$

PowerE: $f_s(s_i, s_j) = e^{d(s_i,s_j)}$

Power10: $f_s(s_i, s_j) = 10^{d(s_i,s_j)}$

combinadas com a distância euclidiana ($d : L_2$). Além disso, compara-se também com a *Slim-tree* sem se fazer nenhuma deformação do espaço.

Para analisar quando a maldição da alta dimensionalidade ocorre nas consultas por similaridade, foram realizadas consultas aos k vizinhos mais próximos ($kNNq$) e consultas por abrangência (Rq) indexando o conjunto sintético de dados com as estruturas *Slim-tree* e a *Slim-tree XS*, analisando seu comportamento conforme a dimensionalidade do conjunto aumenta. O número de vizinhos mais próximos para as consultas $kNNq$ foi $k = 6$ e para as Rq o raio foi escalado para recuperar aproximadamente um por cento de cada conjunto de dimensionalidade diferente, ou seja, é escalado para manter o mesmo o número de elementos retornados.

Os resultados destes experimentos são apresentados na Figura 6.5. Note que os gráficos de ambas consultas $kNNq$ (graficos a),b),c) na linha superior) e Rq (gráficos d),e),f) na linha inferior) seguem o mesmo padrão. Considerando-se o número de cálculos de distância, as Figuras 6.5b) e 6.5e) mostram que o espaço mapeado atenua os efeitos da maldição da alta dimensionalidade. De fato, após 16 dimensões, o espaço mapeado provê uma redução estável de aproximadamente 40% do número de cálculos de distância para se realizar ambos tipos de consultas, independente da base da função de deformação utilizada. Pode-se ver também que, embora o aumento da base da função de deformação reduza o número de cálculos de distância, uma base com valor pequeno já é o suficiente para prover boa redução, e a partir do valor de Euler e, as reduções são relativamente pequenas.

Analisando-se o número de acessos a disco realizados, as Figuras 6.5a) e 6.5d) mostram que o número de acessos a disco necessários no espaço mapeado é próximo porém sempre menor que a estrutura no espaço original. A distância extra armazenada pela *Slim-tree*

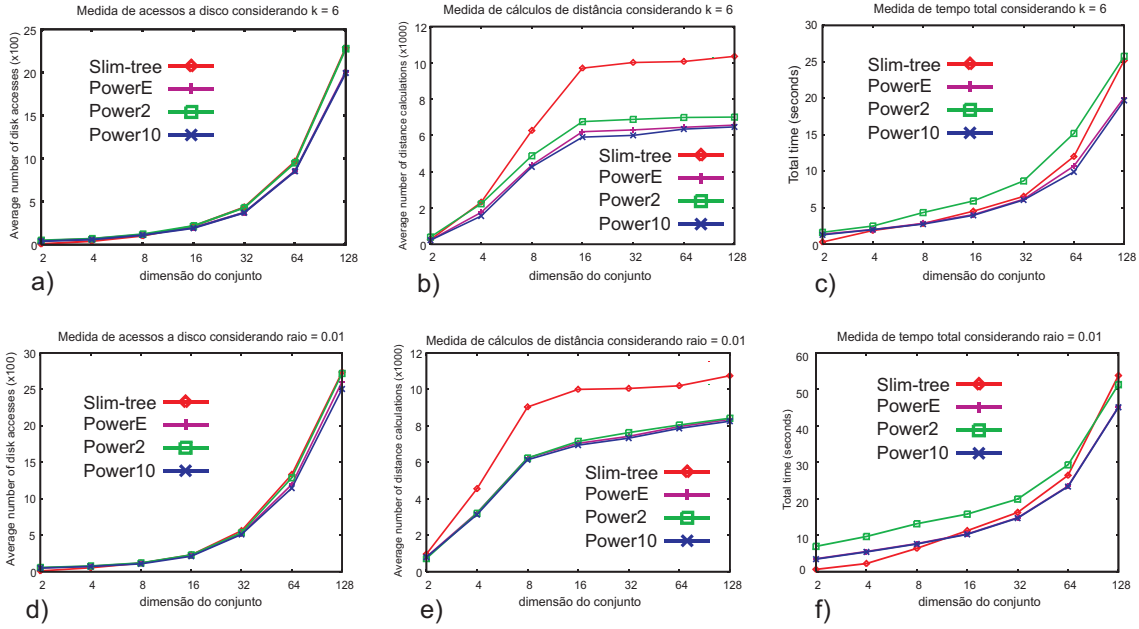


Figura 6.5: Análise comparativa de diferentes funções de deformação sobre consultas por similaridade em conjuntos com aumento gradativo de dimensão. Gráficos a), b), c) correspondem a $kNNq$ e os gráficos d), e), f) correspondem a Rq. Os gráficos a), d) mostram o número médio de acessos a disco, os gráficos b), e) mostram o número médio de cálculos de distância, e os gráficos c), f) mostram o tempo total necessário para se realizarem 500 consultas.

XS, causa um aumento no número de nós na estrutura. Porém, sua maior seletividade nas consultas, atenua este efeito, que no final leva a *Slim-tree XS* realizar menos acessos a disco nos testes.

As Figuras 6.5c) e 6.5f) mostram o tempo gasto para executar as consultas. Os gráficos mostram que a base da deformação do espaço $b = e$ ou $b = 10$ levam a quase o mesmo tempo, e $b = 2$ leva o maior tempo para execução. Isso ocorre devido ao fato do maior custo computacional para processar o logaritmo na base 2. Também pode ser visto que as consultas no espaço original até quatro dimensões são mais rápidas das consultadas no espaço mapeado, mas a partir de oito dimensões elas são mais rápidas se executadas no espaço mapeado.

Assim, a Figura 6.5 mostra que diferentes funções de deformação produzem diferentes espaços de distâncias com pequenas variações na performance. Embora seja intuitivo que funções b^d com um valor grande para b distribuam melhor as distâncias entre pares de elementos, existe uma penalidade no desempenho baseado no algoritmo que calcula as funções exponenciais e as inversas $\log_b d$. Mesmo com essas considerações, quanto maior o valor da

base, melhor é o espaço gerado, embora valores acima do valor e tendam a produzir ganhos menores. Além disso, valores muito altos para a base da função de deformação podem causar estouro númerico no cálculo pelo compilador. Portanto, este experimento mostra que o valor e para a base é uma boa escolha, sendo utilizado para os experimentos mostrados na sequência.

6.3.2 Desempenho de consultas

O segundo experimento realizado mensura o desempenho das consultas por similaridade sobre os conjuntos de palavras (EnglishWords) e de imagens (PCA). A Tabela 6.1 mostra o número de cálculos de distância e o tempo gasto para construir as árvores *Slim-tree* e *Slim-tree XS* para ambos os conjuntos. Como podemos ver na tabela, a construção da *Slim-tree XS* realizou menos cálculos de distância e apresentou o menor tempo para ambos os conjuntos. Além da construção rápida, a *Slim-tree XS* também produz melhores resultados em consultas, como os experimentos a seguir mostram.

	PCA		EnglishWords	
	Distâncias	Tempo (ms)	Distâncias	Tempo (ms)
Slim-tree	1,209,833	3,640	1,882,419	9,437
Slim-tree XS	1,160,576	2,906	1,605,959	7,906

Tabela 6.1: Estatísticas de construção para conjuntos de dados reais.

Neste experimento também mensuramos o número médio de cálculos de distâncias calculadas, o número médio de acessos a disco e o tempo total gasto para realizar 500 consultas para cada conjunto de dados testado. As Figuras 6.6 e 6.7 mostram os resultados. Novamente, o raio máximo das consultas por abrangência, apresentadas na escala logaritmica nos gráficos, representa o raio real do porcentual de dez por cento dos elementos recuperados na consulta.

Analisando os graficos das consultas executadas no conjunto PCA (as duas primeiras colunas da Figura 6.6), podemos notar que a *Slim-tree XS* alcançou melhor desempenho do que a *Slim-tree* no espaço original. De fato, a *Slim-tree* realizou cerca de três vezes mais cálculos de distância em média do que a *Slim-tree XS* (a primeira linha nos gráficos da Figura 6.6). Considerando o número de acessos a disco, os dois tipos de consultas apresentaram

o mesmo comportamento, sendo que a *Slim-tree* realizou mais que o dobro de acessos a disco, em média, que a *Slim-tree XS* (segunda linha da Figura 6.6). Considerando o tempo gasto, a *Slim-tree XS* também teve melhor desempenho que a *Slim-tree*, sendo, em média, o dobro mais rápida que a *Slim-tree*. O mesmo comportamento é visto nos gráficos dos outros conjuntos.

Outro resultado interessante é o desempenho da *Slim-tree XS* indexando o conjunto de palavras inglesas (*EnglishWords*). Este conjunto é adimensional, e a métrica L_{Edit} retorna valores discretos (inteiros) que variam entre zero (identidade) até 24 (tamanho da maior palavra do conjunto). Novamente, podemos ver que a *Slim-tree XS* teve melhor desempenho, uma vez que a *Slim-tree* processou em média 3,5 mais cálculos de distância e o dobro do número de acessos a disco nas consultas. Na medida de tempo, a *Slim-tree XS* se mostrou até 2,5 vezes mais rápida nas consultas.

6.3.3 Escalabilidade

O último experimento mede a escalabilidade da técnica proposta. O experimento consistiu em realizar consultas Rq e $kNNq$ no conjunto sintético (*Synthetic*) de 16 dimensões, aumentando-se sua cardinalidade (número de elementos no conjunto) gradualmente, visando comparar a *Slim-tree XS* com a *Slim-tree*. Foram realizadas consultas usando diferentes funções de deformações, coletando o número médio de acessos a disco, o número médio de cálculos de distância e o tempo total necessário para executar 500 consultas.

A Figura 6.8 mostra os resultados das medidas de desempenho de consultas Rq e $kNNq$ para as diferentes cardinalidades do conjunto sintético de dados. Os graficos mostram que os métodos mostram escala linear com o tamanho do conjunto, independentemente da função de deformação aplicada. Novamente, surge a constatação de que a utilização da base $b = e$ é suficiente para se obter um ganho significativo no desempenho.

6.4 Conclusão

Muitas aplicações lidam com dados complexos, onde um grande número de características (dimensões do vetor) são necessárias para representar a essência dos dados. Assim, estes

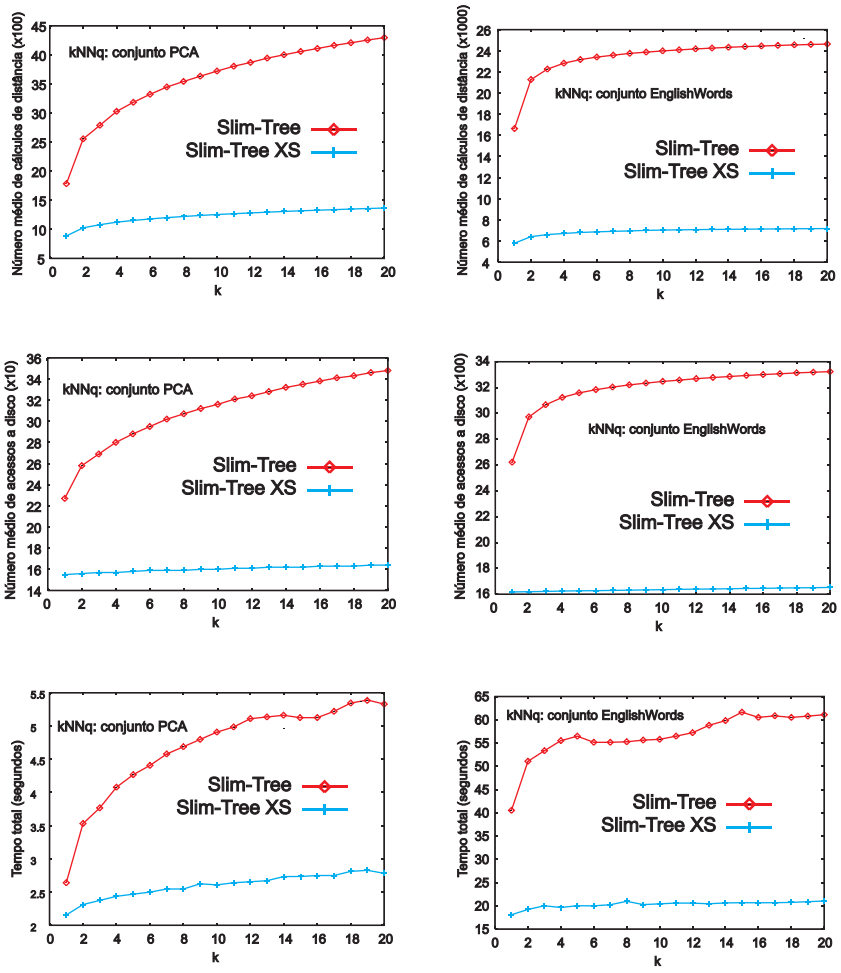


Figura 6.6: Medidas de desempenho para consultas $kNNq$ sobre os conjuntos PCA e EnglishWords.

dados são inerentemente de alta dimensão, que leva ao problema da maldição da alta dimensionalidade na recuperação de dados por similaridade. Consequentemente, novas técnicas são necessárias para atenuar os efeitos deste problema, o que prejudica o desempenho da recuperação dos dados. Trabalhos que visam a reduzir o número de características necessárias para representar o dado foram propostos, mas a maioria deles modifica os atributos do espaço, distorcendo o espaço para a indexação e recuperação. Assim, eles não são adequados para indexação, uma vez que as distorções nas respostas não são facilmente previsíveis.

Este capítulo mostrou uma técnica de mapeamento que exige poucas mudanças nos métodos de acesso métrico, mas reduz grandemente os efeitos negativos da maldição da alta dimensionalidade. A ideia principal é deformar o espaço métrico segundo uma função de deformação, de forma a posicionar elementos que estão longe do centro de consulta ainda

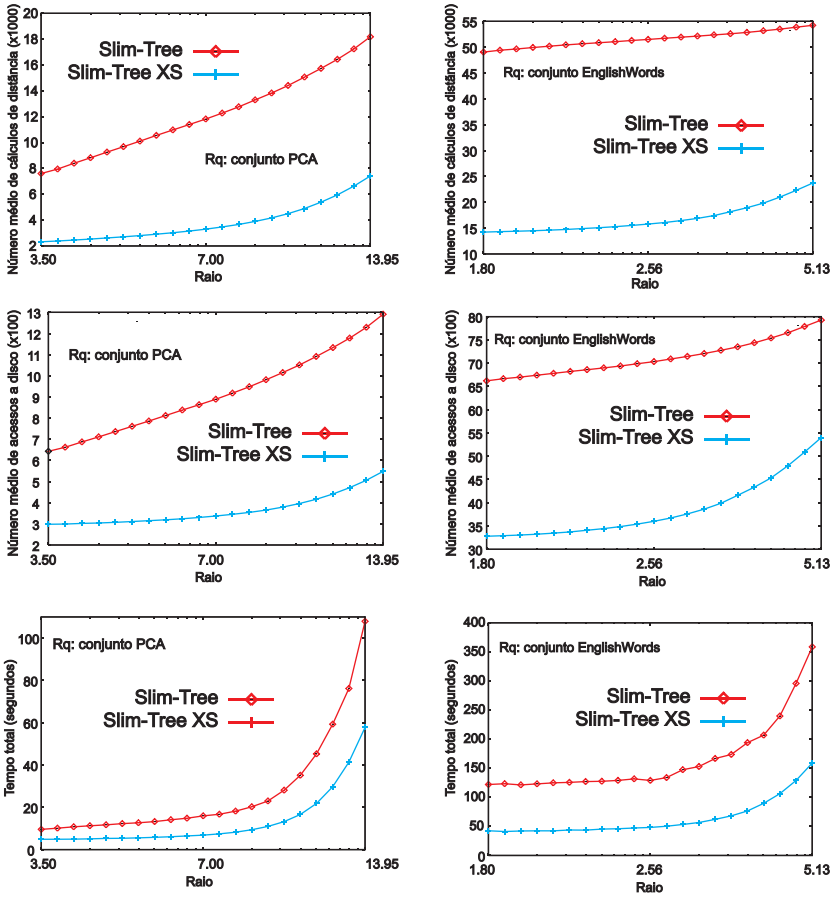


Figura 6.7: Medidas de desempenho para *Rq* sobre os conjuntos PCA e EnglishWords.

mais longe, e os elementos que estão próximos ainda mais perto. Como a função de deformação modifica o espaço das distâncias entre pares de elementos, novas propriedades são derivadas dos axiomas dos espaços métricos. Eles são usados para auxiliar no descarte de subárvores durante as consultas por similaridade.

Experimentos foram realizados para medir a execução das consultas no espaço mapeado utilizando o método de acesso *Slim-tree*. Entretanto, o método proposto pode ser diretamente aplicado a qualquer MAM. A reformulação da *Slim-tree* para trabalhar com o espaço mapeado foi chamada *Slim-tree XS*. Experimentos realizados em conjuntos sintéticos de até 128 dimensões sem correlações entre os atributos mostraram que a *Slim-tree XS* alcançou melhor desempenho que a *Slim-tree* conforme a cardinalidade do conjunto aumenta. Resultados das medidas de desempenho em consultas $kNNq$ mostram que a *Slim-tree XS* realizou até 43% menos cálculos de distância e até 7% menos acessos a disco, resultando em um ganho de tempo de 20%. Resultados das medidas de desempenho de *Rq* mostram que a *Slim-tree*

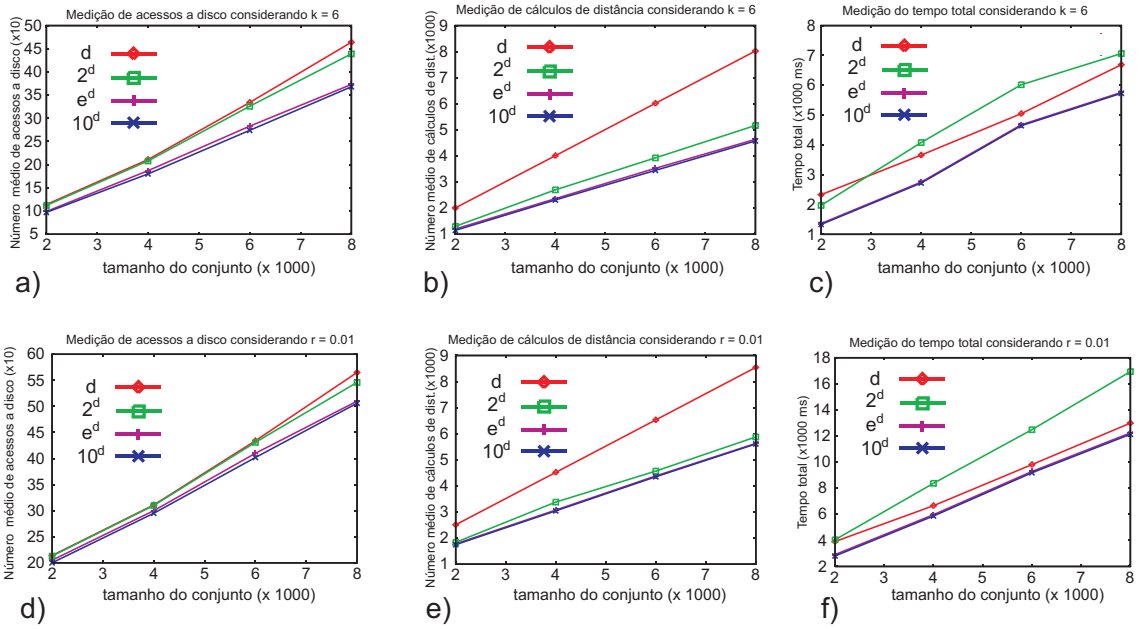


Figura 6.8: Comportamento de diferentes funções de deformação em consultas por similaridade, aplicadas em conjuntos sintéticos com aumento gradativo do número de elementos. A primeira linha corresponde a consultas do tipo $kNNq$ e a segunda a Rq. A primeira coluna mostra o número médio de acessos a disco, a segunda mostra o número médio de cálculos de distância, e a terceira mostra o tempo total necessário para realizar 500 consultas.

XS realizou até 25% menos calculos de distância e até 4% menos acessos a disco, resultando em um ganho de tempo de 18%.

Experimentos realizados em conjuntos reais mostraram que a técnica de mapeamento melhora o desempenho de consultas por similaridade em conjuntos reais de alta dimensão, tal como características extraídas de imagens a partir de métodos PCA e até em conjuntos adimensionais, tais como o conjuntos de palavras da lingua inglesa. Considerando ambos os conjuntos, resultados das medidas de desempenho de $kNNq$ mostram que a *Slim-tree XS* realizou até 70% menos cálculos de distância, 52% menos acessos a disco num ganho de tempo de 57% quando comparada à *Slim-tree* no espaço de distâncias original. Resultados das medidas de desempenho em Rq mostram que a *Slim-tree XS* realizou até 58% menos cálculos de distância, 45% menos acessos a disco num ganho de tempo de 51%. Finalmente, uma análise de escalabilidade realizada em conjuntos sintéticos mostraram que a técnica tem desempenho com comportamento linear conforme aumenta-se o tamanho do conjunto de dados indexado.

Capítulo

7

Conclusão

Esta tese focou o estudo do espaço métrico e suas aplicações na recuperação por conteúdo por similaridade. Ela reúne Teoria dos Espaços Métricos e métodos de acesso métrico de modo que estas duas áreas de pesquisa possam operar em conjunto. Num primeiro instante, foi realizado o embasamento teórico das duas áreas, utilizando conceitos tais como funções de distância e algumas propriedades e definições da Teoria dos Espaços Métricos. O foco principal foi identificar propriedades que auxiliem na tarefa dos operadores de busca por similaridade, de modo a melhorar seu desempenho nas consultas.

A Teoria dos Espaços Métricos possui uma quantidade alta de propriedades e aplicações que podem ser úteis na recuperação por similaridade em dados complexos. Porém, mais importante que resolver problemas específicos, é primordial que a solução encontrada para um dado problema seja genérica o suficiente para ser aplicada em outros problemas da área.

7.1 Principais contribuições desta tese

Esta tese mostrou que algumas aplicações da Teoria dos Espaços Métricos são muito úteis para agilizar os métodos de acesso métrico. Novas técnicas foram desenvolvidas para auxiliar a resolução de problemas que são desafio na área: aplicações lipschitzianas como técnicas de mapeamento e imersões de espaços métricos para o uso de propriedades específicas de cada espaço.

Foi estudada a maldição da dimensionalidade e foi desenvolvida uma técnica para realizar um mapeamento de espaços métricos que levou à atenuação deste efeito, a partir de uma aplicação lipschitziana real baseada em uma função de deformação do espaço das distâncias entre os elementos do conjunto. Foi mostrado que uma função do tipo exponecial deforma as distâncias de modo a diminuir os efeitos da maldição da alta dimensionalidade, melhorando assim o desempenho nas consultas.

Uma segunda contribuição desta tese é uma nova técnica para a imersão de espaços métricos, que pode ser realizada de maneira a preservar as distâncias, possibilitando a utilização de propriedades no espaço de imersão. Para isso foi utilizada a definição do hiperplano métrico. A imersão de espaços métricos no $\mathbb{R}^n$ possibilitou a utilização da lei dos cossenos e assim viabiliza o cálculo de distâncias entre elementos e um hiperplano generalizado, permitindo assim que o hiperplano métrico possa ser utilizado para responder com mais agilidade à consultas por similaridade incluindo kNN e consultas por abrangência. O uso do hiperplano métrico foi exemplificado construindo uma árvore binária métrica, e também foi aplicado em um método de acesso métrico, a família MMH de métodos de acesso métrico, melhorando o particionamento do espaço dos dados.

No período de doutoramento, o candidato estudou propriedades do espaço métrico, e criou algumas técnicas computacionais utilizando-se destas propriedades, propostas para melhorar o desempenho de métodos de acesso métrico. Além disso, propôs novas estruturas de indexação, trabalhou em conjunto com outros pesquisadores na área e cooperou em áreas de pesquisa que necessitam do ferramental computacional existente.

7.2 Trabalhos Futuros

A partir dos métodos desenvolvidos nessa tese, novas aplicações da Teoria dos Espaços Métricos podem ser utilizadas em outros problemas na área de recuperação de dados complexos por similaridade. A seguir são apresentadas algumas ideias para a continuação deste trabalho.

1. Aplicação do mapeamento de espaços métricos para a deformação do espaço das distâncias na família de árvores MMH. A técnica de mapeamento pode ser aplicada na

família MMH de estruturas métricas a fim de se explorar sua eficiência em bases de dados de alta dimensionalidade.

2. Aplicação do hiperplano métrico. Uma vez definida a distância ao hiperplano a partir da imersão de espaços métricos, pode-se utilizar o hiperplano métrico em outros problemas que envolvam dividir o espaço métrico para agilizar consultas.

3. Busca por subsequências de proteinas utilizando MAM com L_{Edit} modificada. A busca por subsequências genéticas, como em bases de proteínas, necessita de busca por subsequências. Pode-se modificar a métrica L_{Edit} para se realizar tais consultas. Porém, faz-se a ressalva de que uma vez indexadas sequências completas, a busca por parte delas usando esta distância faz com que ela deixe de ser métrica, inviabilizando o uso de MAM. Mas este problema pode ser resolvido estudando o limite superior desta função no domínio específico e aplicando-o na desigualdade triangular.

4. Indexação de funções de distância não métricas em MAM. Existem muitas funções de distância não-métricas, e poucos trabalhos proporcionam consultas exatas ao utilizá-las em MAM. A ideia é estudar mecanismos de correção local da desigualdade triangular na estrutura de dados, de modo a podar de modo correto subárvores durante as consultas.

7.3 Publicações

A produção de artigos durante o doutoramento, publicados em conferências internacionais e as publicações em andamento são mencionados a seguir.

1. **Pola, I.R.V., Traina, A.J.M., e Jr., C. T. Distance functions association for content-based image retrieval using multiple comparison criteria. Em CBMS: Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems**. Nesse artigo é apresentada uma técnica de projeção de características que possibilita a indexação de dados em uma estrutura métrica na qual os dados pertencem a diferentes classes e assim possuem diferentes características e funções

de distâncias associadas. Dentro de uma única estrutura, a integração destes dados se torna uma ferramenta muito útil para imagens médicas de diferentes classes [Pola et al., 2006].

2. **Pola, I.R.V., Traina, Jr., C.,e Traina, A.J. The mm-tree: A memory-based metric tree without overlap between nodes. Em ADBIS: Proceedings of the 11th East European conference on Advances in Databases and Information Systems**. Nesse artigo a MAM *MM-tree* foi criada para indexar dados complexos em memória para agilizar consultas por similaridade. A estrutura divide o espaço baseando-se em "bolas" que se intersectam no espaço métrico e possui técnicas que controlam o balanceamento da estrutura, além de algoritmos específicos para consultas que utilizam a proximidade das regiões como ordem de visita dos nós em cada nível da árvore. Deste modo, tem a vantagem de dividir o espaço métrico de modo efetivo e seu desempenho supera as várias estruturas comparadas já existentes na literatura [Pola et al., 2007].

4. **Pola, I.R., Traina, A.J., e Traina, Jr., C. Easing the dimensionality curse by stretching metric spaces. Em SSDBM: Proceedings of the 21st International Conference on Scientific and Statistical Database Management**. As aplicações Lipchitzianas foram utilizadas como técnica de mapeamento de espaços métricos. Um mapeamento que preserva a ordem das distâncias foi realizado através da aplicação de uma função de deformação do espaço das distâncias do Espaço Métrico. Com o mapeamento, um novo espaço foi criado, e portanto, novas propriedades são derivadas para este espaço. Estas novas propriedades são utilizadas em MAM para aumentar a eficiência dos algoritmos de busca na estrutura. Foi mostrado também que os efeitos da maldição da alta dimensionalidade foram atenuados neste novo espaço, proporcionando assim melhor eficiência nos métodos de acesso métrico. [Pola et al., 2009].

5. **Carélo, C.C., Pola, I.R.V., Ciferri, R.R., Traina, A.J., Traina-Jr., C., e Aguiar Ciferri, C.D. The onion-tree: Quick indexing of complex data in the main memory. Em ADBIS: Proceedings of the 13th East European Conference on Advances in Databases and Information Systems**. A *Onion-*

tree, é uma extensão da estrutura *MM-tree*, que foi construída com novas políticas de construção e particionamento do espaço. Ela divide o espaço em regiões baseando-se na multiplicidade do raio dos pivôs iniciais da *MM-tree*, produzindo assim mais regiões de abrangência do que a *MM-tree*. Isso leva a um maior particionamento do espaço métrico e portanto a um melhor desempenho em consultas por similaridade. Este trabalho foi convidado para ser publicado em periódico internacional, e sua extensão e submissão já foi realizada (Elsevier Information Systems) [Carélo et al., 2009].

6. **Barbosa, F.P., Pola, I.R.V., Mangiavacchi, N., e Castelo, A. A numerical method for solving three-dimensional incompressible fluid flows for hydro eletric reservoir applications. Em COBEM: Proceedings of the 20st International Congress of Mechanical Engineering**. Um método numérico foi proposto para simular escoamentos em reservatórios para investigar os efeitos da preservação ambiental. As equações de Navier-Stokes tridimensionais foram utilizadas como equações-base escritas na formulação Euleriana e discretizadas utilizando-se a técnica dos elementos finitos. Os termos convectivos da equação foram discretizados utilizando-se o método Semi-Lagrangeano e o sistema linear foi decomposto num esquema LU através de uma projeção discreta e resolvido com um método iterativo. A malha que representa o domínio foi indexada por uma estrutura de dados topológica construída especificamente para a recuperação de elementos tridimensionais. Os elementos representados pela estrutura de dados são prismas lineares. [Barbosa et al., 2009].

7. Imersões de espaços métricos no $\mathbb{R}^n$. A imersão exata de espaços métricos em outros espaços possibilitou a utilização de propriedades inerentes do espaço mapeado, como a lei dos cossenos em espaços métricos, de modo que a transformação de espaços não cause perda de similaridades ou distorção na ordem das distâncias. Essa propriedade gerou uma nova estrutura binária métrica e foi utilizada para melhorar a divisão do espaço quando aplicada em outros métodos de acesso métricos. Este trabalho está finalizado e encontra-se em fase de submissão para periódico internacional.

Apêndice

A

O MAM Slim-tree

Durante o desenvolvimento do trabalho, foi bastante utilizado o MAM *Slim-tree* nos experimentos. Este apêndice é destinado aos leitores que não conhecem a *Slim-tree* e seu funcionamento. Portanto este apêndice é opcional para os leitores que conhecem a estrutura.

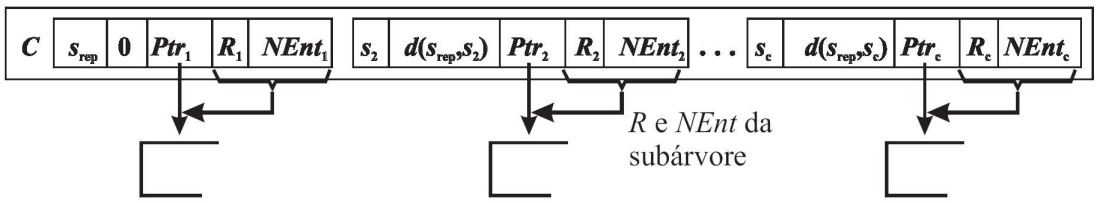
O MAM *Slim-tree* [Traina et al., 2000] usa uma árvore multivias balanceada pela altura, dinâmica, e que tem todos os elementos armazenados nas folhas. A interseção das regiões definidas pelos nós de um mesmo nível desse MAM pode não ser vazia. Ou seja, a divisão do espaço métrico não gera regiões disjuntas, admitindo sobreposição. A estrutura tem ainda recursos para a avaliação do grau de sobreposição entre seus nós (*Fat-Factor*) e um algoritmo de pós-otimização da árvore (*Slim-Down*).

A.1 Estrutura

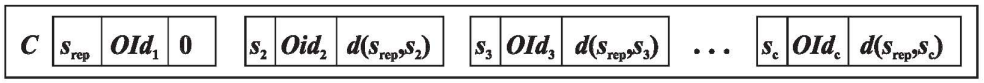
A *Slim-tree* agrupa os objetos do conjunto em páginas de tamanho fixo que representam os nós da árvore. Os agrupamentos são construídos em torno de objetos especialmente selecionados, chamados de elementos representantes, que definem os centros de regiões delimitadas por um raio de cobertura. Os objetos podem ser associados ao nó do elemento representante se eles forem cobertos pelo raio de cobertura associado ao elemento representante em questão. Dessa forma, esta organização permite minimizar tanto o número de acessos a disco

(leituras de páginas) quanto o número de cálculos de distância necessários para responder as consultas.

Os nós da *Slim-tree* podem ser de dois tipos: nó índice (*indexNode*); e nó folha (*leafNode*), que possuem cardinalidade igual ao seu grau. Todos os nós da estrutura, exceto o nó raiz, possuem um objeto representante e um raio de cobertura que engloba todas as suas subárvores. A Figura A.1 ilustra a estrutura lógica do nó índice e folha da *Slim-tree*.



(a) Nó índice



(b) Nó folha

Figura A.1: Estrutura lógica dos nós (a) índice (*indexNode*) e (b) folha (*leafNode*) da *Slim-tree* [C. Traina et al., 2002].

Cada entrada no nó folha é composta pelos atributos que compõe o elemento (s_i); pelo identificador do elemento (OId_i); e pelo valor da distância entre este objeto e o representante do nó ($d(s_{rep}, s_i)$). Essa estrutura pode ser representada como:

$$\text{leafNode [vetor [1..C] de} < s_i,\ OId_i,\ d(s_{rep}, s_i) >]$$

Cada entrada em um nó índice é composta pelo elemento (s_i) que é o representante (centro) da subárvore apontada por Ptr_i; pelo valor da distância entre este elemento e o representante do nó ($d(s_{rep}, s_i)$); por uma ligação para sua subárvore (Ptr_i); um raio de cobertura abrangendo toda a sua subárvore (R_i); e pelo número de entradas ($NEnt_i$) presentes na subárvore apontada por Ptr_i. Essa estrutura pode ser representada como:

$$\text{indexNode [vetor [1..C] de} < s_i,\ d(s_{rep}, s_i),\ Ptr_i,\ R_i,\ NEnt_i >]$$

A Figura A.2 ilustra a estrutura lógica de uma *Slim-tree* contendo sete cadeias de caracteres, cuja função de distância é a L_{Edit}.

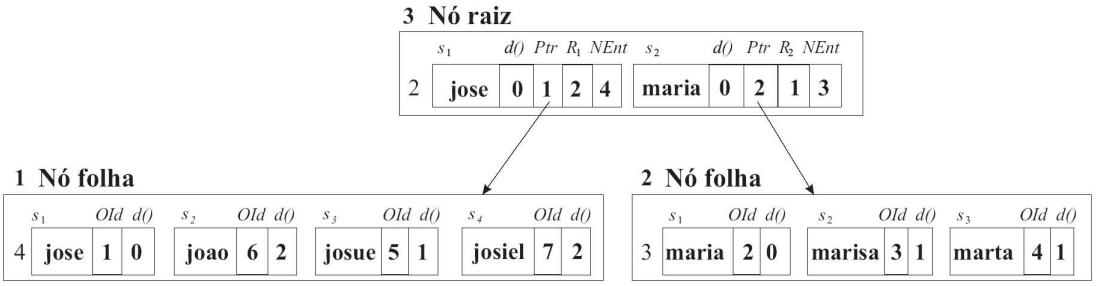


Figura A.2: Exemplo de uma *Slim-tree* indexando cadeias de caracteres com a função de distância L_{Edit}.

A.2 Construção

A *Slim-tree* é construída do nível folha em direção ao nível raiz (*bottom-up*), igual à *B-tree*. Como a *Slim-tree* é um MAM dinâmico, não é necessário que todos os elementos estejam presentes para a construção da árvore. Novos objetos podem ser inseridos depois da estrutura já construída.

O algoritmo de inserção começa a partir do nó raiz, percorrendo a estrutura até encontrar um nó folha, onde a inserção do novo elemento realmente ocorre. O algoritmo tenta localizar um nó que possa receber o novo elemento sem a necessidade de aumento do raio de cobertura. Se mais de um nó pode receber o novo elemento, o algoritmo *ChooseSubtree* é executado para selecionar em qual deles será inserido. Se não existir nenhum nó que possa receber o novo elemento nestas condições, é selecionado um centro que está mais próximo do novo objeto. Esse processo é aplicado recursivamente para todos os níveis da árvore até chegar em um nó folha.

A *Slim-tree* possui três opções para o algoritmo *ChooseSubtree*:

- Aleatório (*random*): seleciona aleatoriamente um dos nós que podem receber o novo elemento sem aumento do raio de cobertura;
- Distância Mínima (*minDist*): seleciona, entre os nós que podem receber o novo elemento, aquele cuja distância entre seu representante (centro) e o novo elemento for mínima;
- Ocupação Mínima (*minOccup*): seleciona, entre os nós que podem receber o novo elemento, aquele cujo número de elementos armazenados seja mínimo. Isso é verificado

através do atributo N_{Ent}. Esta opção é a padrão para a *Slim-tree*.

A escolha da opção para o algoritmo *ChooseSubTree* influencia bastante as características da estrutura resultante. A opção *minOccup* do algoritmo *ChooseSubtree* tende a gerar árvores com maior taxa de ocupação dos nós, como explicitado em [Traina et al., 2000], resultando em árvores mais baixas, porém ao custo de um maior grau de sobreposição. A opção *minDist* tende a criar estruturas com uma taxa de ocupação menor dos nós e com um grau menor de sobreposição, consequentemente as estruturas resultantes são mais altas. Surpreendentemente, a opção *random* consegue gerar estruturas quase tão boas quanto as outras duas, porém sua eficiência e resultados não são determinísticos. Este fenômeno pode indicar que o verdadeiro responsável pelas características da árvore seja mesmo o algoritmo de redistribuição dos objetos entre os nós.

A Figura A.3 apresenta a disposição de 17 objetos $\{s_1, ..., s_{17}\}$ de uma *Slim-tree* com capacidade máxima de três elementos por nó. Os elementos $\{s_1, s_4, s_{10}\}$ estão em um mesmo nó folha e o elemento representante deste nó é o s_1, por isto ele aparece no nó índice do nível superior. Os círculos brancos da Figura A.3(a) representam nós folha, e o de cor cinza os nós índice, onde o tamanho é proporcional ao raio de cobertura e as regiões de sobreposição são indicadas pela intersecção de dois ou mais círculos. Os pontos pretos da Figura A.3(b) indicam os elementos representantes do respectivo nó. Note que o nó raiz não possui nenhum elemento representante.

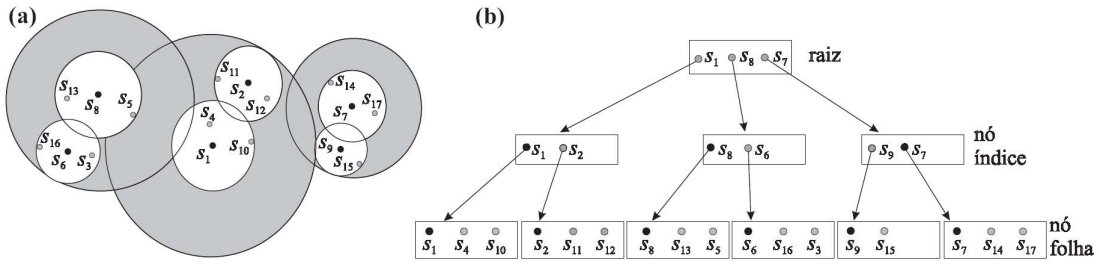


Figura A.3: Representação de uma *Slim-tree* contendo 17 objetos em (a), e sua correspondente estrutura lógica em (b).

Durante o processo de inserção de um novo elemento, pode ocorrer que o nó escolhido para inserção já esteja com a sua taxa de ocupação máxima preenchida. Neste caso, deve ser alocado um novo nó e as entradas que estavam no nó anterior (com a taxa de ocupação preenchida) devem ser redistribuídos entre os dois nós, respeitando a taxa de ocupação

mínima dos nós. Os dois elementos escolhidos como representantes dos dois nós (novo e antigo) e seus respectivos raios de cobertura devem ser inseridos no nó pai. O atributo N_{Ent} também deve ser atualizado nos nós dos níveis superiores. Este processo de divisão, se necessário, pode ser repetido árvore acima. No pior caso, o processo de divisão propaga até o nó raiz e a árvore aumenta a sua altura em uma unidade.

A *Slim-tree* possui três políticas para efetuar a redistribuição dos elementos entre os dois nós, sendo elas:

- **Aleatório** (*random*): seleciona aleatoriamente dois objetos representantes, um para cada nó (novo e antigo), e os demais objetos são distribuídos entre os dois nós pela menor distância entre o elemento e o representante do nó. É o mais rápido, porém seus resultados são os menos satisfatórios;

- **Mínimo dos Maiores Raios** (*minMax*): cada possível par de elementos são considerados candidatos a representantes dos dois nós. Para cada par possível, os demais elementos são inseridos no nó cuja distância ao respectivo representante for a menor. Serão escolhidos como representantes o par de elementos que minimizar o raio de cobertura, ou seja, o menor dos maiores raios de cobertura. A complexidade deste algoritmo é O(C^3), onde C é a cardinalidade do nó, pois para cada elemento escolhe-se dentre os demais quem será seu par, e para cada par possível, verifica-se em qual dos nós centrados em um dos elementos do par cada elemento restante será armazenado. Apesar dele ser extremamente custoso, ele é apontado em [Ciaccia et al., 1998] como o que consegue obter árvores que possibilitam consultas mais eficientes;

- ***Minimal Spanning Tree*** (MST): baseado na *Minimal Spanning Tree* de Kruskal [Kruskal, jr., 1956]. A árvore de caminho mínimo é construída e um dos arcos mais longos da árvore é removido. Dessa forma, dois agrupamentos são obtidos e cada um é associado a um nó. O elemento representante é o elemento central de cada agrupamento. Este algoritmo possibilita a construção de árvores praticamente equivalentes às construídas utilizando o algoritmo de quebra de nós minMax, porém com custo de processamento bem menor, resultando em um menor tempo total de construção. Isso ocorre devido à complexidade do algoritmo ser $O(C^2 logC)$. Esta opção é a padrão para

a *Slim-tree* redistribuir os elementos. A Figura A.4 ilustra o mecanismo de quebra de nós utilizando o algoritmo MST. Em (a) é representada a disposição dos elementos antes da divisão; em (b) é ilustrada a árvore de caminho mínimo para os elementos do nó corrente; já em (c) é ilustrada a nova distribuição para os dois novos nós.

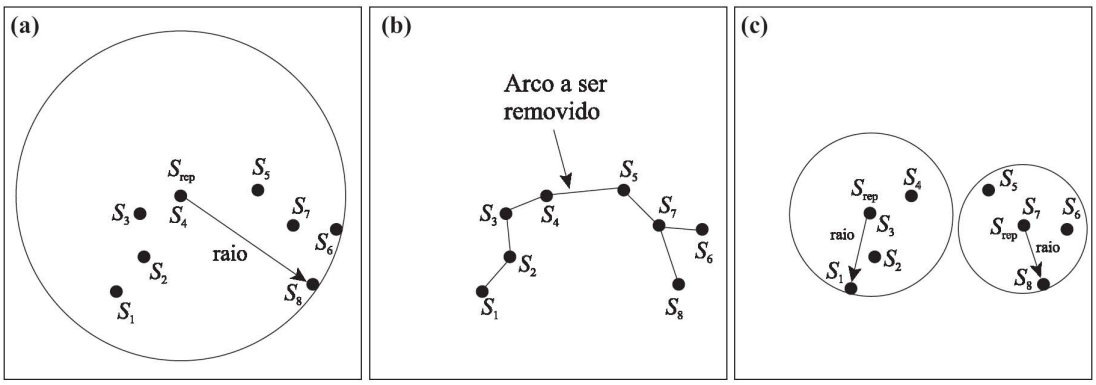


Figura A.4: Exemplificação do mecanismo de quebra de nós segundo a política MST (*Minimal Spanning Tree*) em um conjunto de elementos: (a) estrutura do nó antes da divisão;(b) MST construída sobre os elementos do nó; (c) organização dos elementos depois da divisão.

Na opção MST não existe garantia que os nós resultantes da divisão tenham ocupação mínima de entradas, podendo até existir nós com apenas uma única entrada (apenas o elemento representante com raio nulo para os nós folha). Nestes casos, a probabilidade de inserção de novos elementos, para a opção minDist do algoritmo *ChooseSubtree*, é baixa devido ao raio do elemento representante ser nulo.

A.3 *FatFactor* e *Slim-Down*

A sobreposição entre os nós das árvores métricas é um efeito indesejável, pois obriga a busca em profundidade em diversas subárvores para a localização dos elementos solicitados pelas consultas. Isso ocorre porque mais nós da árvore serão consultados (todos os que têm região se sobrepondo à região de consulta), e o descarte de subárvores fica prejudicado. Sua influência no desempenho das buscas pode ser até maior que a altura total das árvores.

Visando evitar essa deficiência, a *Slim-tree* foi desenvolvida com o objetivo de diminuir a sobreposição entre os nós da árvore, bem como oferecer mecanismos para verificação da porcentagem de sobreposição existente na árvore. Para isso, em Traina [Traina et al., 2000],

foi definido o fator de sobreposição *Fat-Factor* e o algoritmo de reorganização da árvore *Slim-Down*.

O *Fat-Factor* é uma medida para avaliar o grau de sobreposição entre os nós da árvore. Ela foi usada para desenvolver um algoritmo de reorganização dos nós, chamado *Slim-Down*. Esse algoritmo visa reduzir o grau de sobreposição entre os nós através da troca de objetos apenas entre os nós folha. Os elementos são transferidos de um nó origem para um destino caso o nó destino tenha espãço para o armazenamento da entrada, e ainda, a inserçãoo da entrada não causa aumento do raio de cobertura, podendo diminuir o raio de cobertura do nó origem. Esse processo é realizado até que não seja mais possível realizar trocas de elementos entre os nós envolvidos ou que passe de um certo valor limite de número de trocas. Este processo de reorganização das entradas pode diminuir a sobreposição dos nós e, consequentemente, diminuir o número de acesso a disco para responder a consultas.

Por ser um processo relativamente custoso, é recomendado que o *Slim-Down* seja executado apenas quando o grau de sobreposição, definido através do *Fat-Factor*, ultrapasse um certo limite para toda a árvore, ou que uma grande quantidade de elementos já tenha sido inserida na *Slim-tree*.

Através desses conceitos, o MAM *Slim-tree* permite realizar consultas por similaridade minimizando tanto o número de cálculos de distâncias quanto o de acessos a disco.

Referências Bibliográficas

[Adler & Heeringa, 2008] Adler, M. e Heeringa, B. (2008). Search space reductions for nearest-neighbor queries. In *TAMC'08: Proceedings of the 5th international conference on Theory and applications of models of computation*, p. 554–567, Berlin, Heidelberg. Springer-Verlag.

[Aggarwal, 2005] Aggarwal, C. C. (2005). Towards exploratory test instance specific algorithms for high dimensional classification. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, p. 526–531, New York, NY, USA. ACM.

[Aggarwal et al., 2001] Aggarwal, C. C., Hinneburg, A., e Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional spaces. In Bussche, J. V. d. e Vianu, V., editors, *ICDT*, p. 420–434, London, UK. Springer Verlag.

[Aggarwal & Yu, 2002] Aggarwal, C. C. e Yu, P. S. (2002). Redefining clustering for high-dimensional applications. *IEEE Trans. on Knowl. and Data Eng.*, 14(2):210–225.

[Aronovich & Spiegler, 2007] Aronovich, L. e Spiegler, I. (2007). Cm-tree: A dynamic clustered index for similarity search in metric databases. *Data Knowl. Eng.*, 63(3):919–946.

[Aronovich & Spiegler, 2010] Aronovich, L. e Spiegler, I. (2010). Bulk construction of dynamic clustered metric trees. *Knowl. Inf. Syst.*, 22(2):211–244.

[Ashby & Perrin, 1988] Ashby, F. G. e Perrin, N. A. (1988). Toward a unified theory of similarity and recognition. *Psychological Review*, 95(1):124–150.

[Athitsos et al., 2008] Athitsos, V., Papapetrou, P., Potamias, M., Kollios, G., e Gunopulos, D. (2008). Approximate embedding-based subsequence matching of time series. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, p. 365–378, New York, NY, USA. ACM.

[Baeza-Yates et al., 1994] Baeza-Yates, R. A., Cunto, W., Manber, U., e Wu, S. (1994). Proximity matching using fixed-queries trees. In Crochemore, M. e Gusfield, D., editors, *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, p. 198–212, Asilomar, CA. Springer-Verlag, Berlin.

[Bai et al., 2008] Bai, X., Yang, X., e Latecki, L. J. (2008). Detection and recognition of contour parts based on shape similarity. *Pattern Recogn.*, 41(7):2189–2199.

[Baioco et al., 2007] Baioco, G. B., Traina, A. J. M., e Traina, Jr., C. (2007). An effective cost model for similarity queries in metric spaces. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, p. 527–528, New York, NY, USA. ACM.

[Barbosa et al., 2009] Barbosa, F. P., Pola, I. R. V., Mangiavacchi, N., e Castelo, A. (2009). A numerical method for solving three-dimensional incompressible fluid flows for hydroeletric reservoir applications. In *COBEM2009: Proceedings of the 20st International Congress of Mechanical Engineering*.

[Barioni, 2002] Barioni, M. C. N. (2002). *Operações de consulta por similaridade em grandes bases de dados complexos*. Tese de doutorado, Universidade de São Paulo. Orientador: Caetano Traina Junior.

[Barioni et al., 2006] Barioni, M. C. N., Razente, H., Traina, A., e Traina, Jr., C. (2006). Siren: a similarity retrieval engine for complex data. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, p. 1155–1158. VLDB Endowment.

[Barioni et al., 2009] Barioni, M. C. N., Razente, H. L., Traina, Jr, A. J. M., e Traina, C. (2009). Seamlessly integrating similarity queries in sql. *Softw. Pract. Exper.*, 39(4):355–384.

[Bayer & McCreight, 2002] Bayer, R. e McCreight, E. (2002). Organization and maintenance of large ordered indexes. *Software pioneers: contributions to software engineering*, p. 245–262.

[Belussi & Faloutsos, 1995] Belussi, A. e Faloutsos, C. (1995). Estimating the selectivity of spatial queries using the 'correlation' fractal dimension. In Dayal, U., Gray, P. M. D., e Nishio, S., editors, *Proc. 21st Int. Conf. Very Large Data Bases, VLDB*, p. 299–310. Morgan Kaufmann.

[Berchtold et al., 1997] Berchtold, S., Böhm, C., Keim, D. A., e Kriegel, H.-P. (1997). A cost model for nearest neighbor search in high-dimensional data space. In *ACM Symposium on Principles of Database Systems (PODS'1997)*, Tucson, AZ. ACM Press.

[Berchtold et al., 1996] Berchtold, S., Keim, D. A., e Kriegel, H.-P. (1996). The x-tree: An index structure for high-dimensional data. In *Intl. Conf. on Very Large Databases (VLDB)*, p. 28–39, Bombay, India.

[Bergroth et al., 2000] Bergroth, L., Hakonen, H., e Raita, T. (2000). A survey of longest common subsequence algorithms. In *SPIRE '00: Proceedings of the Seventh International Symposium on String Processing Information Retrieval (SPIRE'00)*, page 39, Washington, DC, USA. IEEE Computer Society.

[Beyer et al., 1999a] Beyer, K., Godstein, J., Ramakrishnan, R., e Shaft, U. (1999a). When is "nearest neighbor"meaningful? In Beeri, C. e Buneman, P., editors, *International Conference on Database Theory (ICDT)*, v. 1540 of *Lecture Notes in Computer Science*, p. 217–235, Jerusalem, Israel. Springer Verlag.

[Beyer et al., 1999b] Beyer, K. S., Goldstein, J., Ramakrishnan, R., e Shaft, U. (1999b). When is "nearest neighbor" meaningful? In *ICDT '99: Proceeding of the 7th International Conference on Database Theory*, p. 217–235. Springer-Verlag.

[Böhm, 2000] Böhm, C. (2000). A cost model for query processing in high dimensional data spaces. *ACM Transactions on Database Systems (TODS)*, 25(2):129 – 178.

[Biederman, 1987] Biederman, I. (1987). Recognition-by-components: a theory of human image understanding. *Psychological Review*, 94(2):115–147.

[Blum & Langley, 1997] Blum, A. L. e Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271.

[Bookstein et al., 2002] Bookstein, A., Kulyukin, V. A., e Raita, T. (2002). Generalized hamming distance. *Inf. Retr.*, 5(4):353–375.

[Bozkaya & Ozsoyoglu, 1997] Bozkaya, T. e Ozsoyoglu, M. (1997). Distance-based indexing for high-dimensional metric spaces. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, p. 357–368. ACM Press.

[Bozkaya & Ozsoyoglu, 1999] Bozkaya, T. e Ozsoyoglu, M. (1999). Indexing large metric spaces for similarity search queries. *ACM Transactions on Database Systems*, 24(3):361–404.

[Brin, 1995] Brin, S. (1995). Near neighbor search in large metric spaces. In Dayal, U., Gray, P. M. D., e Nishio, S., editors, *Intl. Conf. on Very Large Databases (VLDB)*, p. 574–584, Zurich, Switzerland. Morgan Kaufmann.

[Bueno et al., 2008] Bueno, R., Santos Kaster, D., Traina, A. J., e Traina, Jr., C. (2008). A new approach for optimization of dynamic metric access methods using an algorithm of effective deletion. In *SSDBM '08: Proceedings of the 20th international conference on Scientific and Statistical Database Management*, p. 366–383, Berlin, Heidelberg. Springer-Verlag.

[Bugatti et al., 2008] Bugatti, H. P., Traina, A. J. M., e Traina, Caetano, J. (2008). Assessing the best integration between distance-function and image-feature to answer similarity queries. In *23rd Annual ACM Symposium on Applied Computing (SAC2008)*, p. 1225–1230, Fortaleza, Ceará - Brazil. ACM Press.

[Burkhard & Keller, 1973] Burkhard, W. A. e Keller, R. M. (1973). Some approaches to best-match file searching. *Communications of the ACM*, 16(4):230–236.

[C. Traina et al., 2002] C. Traina, J., Traina, A., Faloutsos, C., e Seeger, B. (2002). Fast indexing and visualization of metric data sets using slim-trees. *IEEE Transactions on Knowledge and Data Engineering*, 14(2):244–260.

[Carélo et al., 2009] Carélo, C. C., Pola, I. R., Ciferri, R. R., Traina, A. J., Traina-Jr., C., e Aguiar Ciferri, C. D. (2009). The onion-tree: Quick indexing of complex data in the main memory. In *ADBIS '09: Proceedings of the 13th East European Conference on Advances in Databases and Information Systems*, p. 235–252, Berlin, Heidelberg. Springer-Verlag.

[Carson et al., 1997] Carson, C., Belongie, S., Greenspan, H., e Malik, J. (1997). Region-based image querying. In *Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries*, p. 42–49, San Juan, Puerto Rico.

[Chang & Ghosh, 1998] Chang, K. e Ghosh, J. (1998). Principal curves for nonlinear feature extraction and classification. *Applications of Artificial Neural Networks in Image Processing III (SPIE)*, 3307:120–129.

[Chen et al., 2009] Chen, Q., Kotani, K., Lee, F., e Ohmi, T. (2009). A fast retrieval of dna sequences using histogram information. In *FITME '09: Proceedings of the 2009 Second International Conference on Future Information Technology and Management Engineering*, p. 529–532, Washington, DC, USA. IEEE Computer Society.

[Chávez et al., 2001] Chávez, E., Navarro, G., Baeza-Yates, R. A., e Marroquín, J. L. (2001). Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321.

[Ciaccia et al., 1999] Ciaccia, P., Nanni, A., e Patella, M. (1999). A query-sensitive cost model for similarity queries with m-tree. In *Australasian Database Conference*, Auckland, New Zealand.

[Ciaccia et al., 1997] Ciaccia, P., Patella, M., e Zezula, P. (1997). M-tree: An efficient access method for similarity search in metric spaces. In Jarke, M., editor, *Intl. Conf. on Very Large Databases (VLDB)*, p. 426–435, Athens, Greece.

[Ciaccia et al., 1998] Ciaccia, P., Patella, M., e Zezula, P. (1998). A cost model for similarity queries in metric spaces. In *ACM Symposium on Principles of Database Systems (PODS)*, Seattle, Washington. ACM Press.

[Comer, 1979] Comer, D. (1979). Ubiquitous b-tree. *ACM Comput. Surv.*, 11(2):121–137.

[Damerau, 1964] Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176.

[Daoudi et al., 2009] Daoudi, I., Idrissi, K., Ouatik, S. E., Baskurt, A., e Aboutajdine, D. (2009). An efficient high-dimensional indexing method for content-based retrieval in large image databases. *Image Commun.*, 24(10):775–790.

[Datta et al., 2008] Datta, R., Joshi, D., Li, J., e Wang, J. Z. (2008). Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2):1–60.

[DeMers & Cottrell, 1993] DeMers, D. e Cottrell, G. W. (1993). Non-linear dimensionality reduction. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, p. 580–587, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Do & Wang, 2009] Do, H. T. e Wang, D. (2009). Overlap-based similarity metrics for motif search in dna sequences. In *ICONIP '09: Proceedings of the 16th International Conference on Neural Information Processing*, p. 465–474, Berlin, Heidelberg. Springer-Verlag.

[D'Ulizia et al., 2007] D'Ulizia, A., Ferri, F., e Grifoni, P. (2007). Approximate queries by relaxing structural constraints in gis. In *ER'07: Proceedings of the 2007 conference on Advances in conceptual modeling*, p. 398–408, Berlin, Heidelberg. Springer-Verlag.

[D'Yachkov & Voronina, 2009] D'Yachkov, A. G. e Voronina, A. N. (2009). Dna codes for additive stem similarity. *Probl. Inf. Transm.*, 45(2):124–144.

[Euachongprasit & Ratanamahatana, 2008] Euachongprasit, W. e Ratanamahatana, C. A. (2008). Efficient multimedia time series data retrieval under uniform scaling and normalisation. In *ECIR'08: Proceedings of the IR research, 30th European conference on Advances in information retrieval*, p. 506–513, Berlin, Heidelberg. Springer-Verlag.

[Falomir et al., 2009] Falomir, Z., Almazán, J., Grande, J., Museros, L., e Escrig, M. T. (2009). A similarity calculus for comparing qualitative shape descriptions. In *Proceeding of the 2009 conference on Artificial Intelligence Research and Development*, p. 318–326, Amsterdam, The Netherlands, The Netherlands. IOS Press.

[Faloutsos, 1996] Faloutsos, C. (1996). *Searching Multimedia Databases by Content*. Kluwer Academic Publishers, Boston, MA.

[Faloutsos & Kamel, 1994] Faloutsos, C. e Kamel, I. (1994). Beyond uniformity and independence: Analysis of r-trees using the concept of fractal dimension. In *Symposium on Principles of Database Systems (PODS'1994)*, Minneapolis, MN. ACM Press.

[Fanjun et al., 2009] Fanjun, L., Binggang, C., Junping, W., e Kai, Z. (2009). Color image similarity measurement and its application in matching. In *ICICTA '09: Proceedings of the 2009 Second International Conference on Intelligent Computation Technology and Automation*, p. 566–568, Washington, DC, USA. IEEE Computer Society.

[Felipe et al., 2003] Felipe, J. C., Traina, A. J. M., e Jr., C. T. (2003). Retrieval by content of medical images using texture for tissue identification. In *16th IEEE Symposium on Computer-Based Medical Systems (CBMS'03)*, p. 175–180. IEEE Computer Society.

[Ferragina & Grossi, 1999] Ferragina, P. e Grossi, R. (1999). The string b-tree: a new data structure for string search in external memory and its applications. *J. ACM*, 46(2):236–280.

[Ferreira et al., 2005] Ferreira, K. R., Vinhas, L., de Queiroz, G. R., de Souza, R. C. M., e Câmara, G. (2005). The architecture of a flexible querier for spatio-temporal databases. In *GeoInfo*, p. 155–173.

[Ferreira et al., 2009] Ferreira, M. R. P., Traina, A. J. M., Dias, I., Chbeir, R., e Traina Jr., C. (2009). Identifying algebraic properties to support optimization of unary similarity queries. In Arenas, M. e Bertossi, L., editors, *Proceedings of the 3rd Alberto Mendelzon International Workshop on Foundations of Data Management (AMW'09)*, v. 450 of *CEUR Workshop Proceedings*, p. 1–10, Arequipa, Peru. CEUR-WS.

[Fu et al., 2008a] Fu, A. W.-C., Keogh, E., Lau, L. Y., Ratanamahatana, C. A., e Wong, R. C.-W. (2008a). Scaling and time warping in time series querying. *The VLDB Journal*, 17(4):899–921.

[Fu et al., 2008b] Fu, J., Joshi, S. B., e Simpson, T. W. (2008b). Shape differentiation of freeform surfaces using a similarity measure based on an integral of gaussian curvature. *Comput. Aided Des.*, 40(3):311–323.

[Gaede & Günther, 1998] Gaede, V. e Günther, O. (1998). Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231.

[Gal & Toledo, 2005] Gal, E. e Toledo, S. (2005). Algorithms and data structures for flash memories. *ACM Comput. Surv.*, 37(2):138–163.

[Gao et al., 2009a] Gao, Y., Zheng, B., Chen, G., e Li, Q. (2009a). On efficient mutual nearest neighbor query processing in spatial databases. *Data Knowl. Eng.*, 68(8):705–727.

[Gao et al., 2009b] Gao, Y., Zheng, B., Lee, W.-C., e Chen, G. (2009b). Continuous visible nearest neighbor queries. In *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*, p. 144–155, New York, NY, USA. ACM.

[Gharib, 2009] Gharib, T. F. (2009). A hybrid approach for indexing and searching protein structures. *W. Trans. on Comp.*, 8(6):966–975.

[Guliato et al., 2009] Guliato, D., de Melo, E. V., Rangayyan, R. M., e Soares, R. C. (2009). Postgresql-ie: An image-handling extension for postgresql. *J. Digital Imaging*, 22(2):149–165.

[Guttman, 1984] Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *In Proceedings of ACM SIGMOD Conference of Management of Data*, p. 47–57.

[Hamming, 1986] Hamming, R. W. (1986). *Coding and Information Theory*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Haralick et al., 1973] Haralick, R. M., Shanmugam, K., e Dinstein, I. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6):610–621.

[Haykin, 1994] Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan, New York.

[Hitchcock, 1941] Hitchcock, F. L. (1941). The distribution of a product from several sources to numerous localities. *Journal of Math. Phys.*, 20:224–230.

[Hjaltason & Samet, 2003] Hjaltason, G. R. e Samet, H. (2003). Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems (TODS)*, 21:517 – 580.

[Hoksza, 2009] Hoksza, D. (2009). Ddpin: distance and density based protein indexing. In *CIBCB'09: Proceedings of the 6th Annual IEEE conference on Computational Intelligence in Bioinformatics and Computational Biology*, p. 263–270, Piscataway, NJ, USA. IEEE Press.

[Hunt et al., 2001] Hunt, E., Atkinson, M. P., e Irving, R. W. (2001). A database index to large biological sequences. In Apers, P. M. G., Atzeni, P., Ceri, S., Paraboschi, S., Ramamohanarao, K., e Snodgrass, R. T., editors, *27th International Conference on Very Large Data Bases*, p. 139–148, Roma, Italy. Morgan Kaufmann.

[Hyvärinen, 1999] Hyvärinen, A. (1999). Survey on independent component analysis.

[Jian et al., 2008] Jian, M., Chen, S., e Dong, J. (2008). Illumination-invariant texture classification based on self-similarity and gabor wavelet. In *IITA '08: Proceedings of the 2008 Second International Symposium on Intelligent Information Technology Application*, p. 352–355, Washington, DC, USA. IEEE Computer Society.

[Jr. et al., 2000a] Jr., C. T., Traina, A. J. M., e Faloutsos, C. (2000a). Distance exponent: A new concept for selectivity estimation in metric trees. In *International Conference on Data Engineering - ICDE*, page 195, San Diego - CA. IEEE Computer Society.

[Jr. et al., 2000b] Jr., C. T., Traina, A. J. M., e Faloutsos, C. (2000b). Distance exponent: a new concept for selectivity estimation in metric trees. In *Intl. Conf. on Data Engineering (ICDE)*, San Diego - CA. IEEE Computer Society.

[Jr. et al., 2000c] Jr., C. T., Traina, A. J. M., Wu, L., e Faloutsos, C. (2000c). Fast feature selection using fractal dimension. In *Brazilian Symposium on Databases (SBBD)*, João Pessoa - PA, Brazil.

[Junjun et al., 2008] Junjun, H., Jingxiu, Z., e Mian, W. (2008). An approach of color image edge detection based on triangle similarity. In *CCCM '08: Proceedings of the 2008 ISECS International Colloquium on Computing, Communication, Control, and Management*, p. 129–132, Washington, DC, USA. IEEE Computer Society.

[Kahveci & Singh, 2001] Kahveci, T. e Singh, A. K. (2001). Efficient index structures for string databases. In *The VLDB Journal*, p. 351–360.

[Kamel & Faloutsos, 1993] Kamel, I. e Faloutsos, C. (1993). On packing r-trees. In *International Conferenceon Information and Knowledge Management (CIKM)*.

[Kang et al., 2009] Kang, H.-Y., Kim, J.-S., e Li, K.-J. (2009). Similarity measures for trajectory of moving objects in cellular space. In *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, p. 1325–1330, New York, NY, USA. ACM.

[Kaplan et al., 1998] Kaplan, L. M., Murenzi, R., e Namuduri, K. R. (1998). Fast fractal feature extraction for texture segmentation using wavelets. In Sethi, I. K. e Jain, R. C., editors, *Storage and Retrieval for Image and Video Databases VI*, p. 162–173.

[Katayama & Satoh, 2001] Katayama, N. e Satoh, S. (2001). Distinctiveness-sensitive nearest neighbor search for efficient similarity retrieval of multimedia information. In *ICDE*, p. 493–502, Washington, DC, USA. IEEE Computer Society.

[Keogh & Ratanamahatana, 2005] Keogh, E. e Ratanamahatana, C. A. (2005). Exact indexing of dynamic time warping. *Knowl. Inf. Syst.*, 7(3):358–386.

[Keogh et al., 2009] Keogh, E., Wei, L., Xi, X., Vlachos, M., Lee, S.-H., e Protopapas, P. (2009). Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures. *The VLDB Journal*, 18(3):611–630.

[Kim et al., 2006] Kim, S.-W., Yoon, J., Park, S., e Won, J.-I. (2006). Shape-based retrieval in time-series databases. *J. Syst. Softw.*, 79(2):191–203.

[Knuth, 1998] Knuth, D. E. (1998). *The Art of Computer Programming - Sorting and Searching*. Addison-Wesley Professional, 2nd edition.

[Korn & Muthukrishnan, 2000] Korn, F. e Muthukrishnan, S. (2000). Influence sets based on reverse nearest neighbor queries. *SIGMOD Rec.*, 29(2):201–212.

[Korn et al., 2001] Korn, F., Pagel, B.-U., e Faloutsos, C. (2001). On the 'dimensionality curse' and the 'self-similarity blessing'. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):96–111. Elaine Josiel.

[Kramer, 1991] Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243.

[Kruskal, jr., 1956] Kruskal, jr., J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proc. Amer. Math. Soc.*, v. 7, p. 48–50.

[Leung et al., 2008] Leung, C. H., Liu, J., Chan, A. W., e Milani, A. (2008). An architectural paradigm for collaborative semantic indexing of multimedia data objects. In *VISUAL '08: Proceedings of the 10th international conference on Visual Information Systems*, p. 216–226, Berlin, Heidelberg. Springer-Verlag.

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710.

[Levi & Klein, 2000] Levi, D. M. e Klein, S. A. (2000). Seeing circles: what limits shape perception? *Vision Research*, 40(17):2329–2339.

[Lian et al., 2009] Lian, X., Chen, L., Yu, J. X., Han, J., e Ma, J. (2009). Multiscale representations for fast pattern matching in stream time series. *IEEE Trans. on Knowl. and Data Eng.*, 21(4):568–581.

[Lima, 1975] Lima, E. L. (1975). *Espaços Métricos*. Impa.

[Lin, 2008] Lin, H.-Y. (2008). Compressed b+-trees. *W. Trans. on Comp.*, 7(12):2001–2010.

[Liu & Hua, 2009] Liu, D. e Hua, K. A. (2009). Transfer non-metric measures into metric for similarity search. In *MM '09: Proceedings of the seventeen ACM international conference on Multimedia*, p. 693–696, New York, NY, USA. ACM.

[Liu & Yu, 2005] Liu, H. e Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. on Knowl. and Data Eng.*, 17(4):491–502.

[Liu et al., 2007] Liu, Y., Zhang, D., Lu, G., e Ma, W.-Y. (2007). A survey of content-based image retrieval with high-level semantics. *Pattern Recogn.*, 40(1):262–282.

[Loffler et al., 2003] Loffler, G., Wilson, H. R., e Wilkinson, F. (2003). Local and global contributions to shape discrimination. *Vision Research*, 43(5):519–530.

[Lokoc, 2009] Lokoc, J. (2009). Parallel dynamic batch loading in the m-tree. In *SISAP '09: Proceedings of the 2009 Second International Workshop on Similarity Search and Applications*, p. 117–123, Washington, DC, USA. IEEE Computer Society.

[Manjunath & Ma, 1996] Manjunath, B. S. e Ma, W. Y. (1996). Texture features for browsing and retrieval of large image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842.

[Marteau, 2009] Marteau, P.-F. (2009). Time warp edit distance with stiffness adjustment for time series matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):306–318.

[Medeiros et al., 2005] Medeiros, C. B., Egenhofer, M. J., e Bertino, E., editors (2005). *Advances in Spatial and Temporal Databases, 9th International Symposium, SSTD 2005, Angra dos Reis, Brazil, August 22-24, 2005, Proceedings*, v. 3633 of *Lecture Notes in Computer Science*. Springer.

[Melton & Simon, 2001] Melton, J. e Simon, A. R. (2001). *SQL : 1999 - Understanding Relational Language Components*. Morgan Kaufmann, San Francisco, Califórnia.

[Müller et al., 2004] Müller, H., Michoux, N., Bandon, D., e Geissbuhler, A. (2004). A review of content-based image retrieval systems in medical applications-clinical benefits and future directions. *International Journal of Medical Informatics*, 73(1):1–23.

[Montani et al., 2009] Montani, S., Bottrighi, A., Leonardi, G., Portinale, L., e Terenziani, P. (2009). Multi-level abstractions and multi-dimensional retrieval of cases with time series features. In *ICCBR '09: Proceedings of the 8th International Conference on Case-Based Reasoning*, p. 225–239, Berlin, Heidelberg. Springer-Verlag.

[Nadvorny & Heuser, 2004] Nadvorny, C. F. e Heuser, C. A. (2004). Twisting the metric space to achieve better metric trees. In *SBBD*, p. 178–190.

[Ocsa & Cuadros-Vargas, 2007] Ocsa, A. e Cuadros-Vargas, E. (2007). Dbm*-tree: an efficient metric access method. In *ACM-SE 45: Proceedings of the 45th annual southeast regional conference*, p. 401–406, New York, NY, USA. ACM.

[Papadopoulos & Manolopoulos, 1997] Papadopoulos, A. e Manolopoulos, Y. (1997). Performance of nearest neighbor queries in r-trees. In *International Conference on Database Theory (ICDT)*, Delphi,Greece.

[Pola et al., 2009] Pola, I. R., Traina, A. J., e Traina, Jr., C. (2009). Easing the dimensionality curse by stretching metric spaces. In *SSDBM 2009: Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, p. 417–434, Berlin, Heidelberg. Springer-Verlag.

[Pola et al., 2007] Pola, I. R., Traina, Jr., C., e Traina, A. J. (2007). The mm-tree: A memory-based metric tree without overlap between nodes. In *ADBIS '07: Proceedings of the 11th East European conference on Advances in Databases and Information Systems*, p. 157–171, Berlin, Heidelberg. Springer-Verlag.

[Pola et al., 2006] Pola, I. R. V., Traina, A. J. M., e Jr., C. T. (2006). Distance functions association for content-based image retrieval using multiple comparison criteria. In *CBMS '06: Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*, p. 899–904, Washington, DC, USA. IEEE Computer Society.

[Qi et al., 2008] Qi, L., Zhang, L., Dong, J., Yu, Z., e Yang, A. (2008). Self-similarity based classification of 3d surface textures. In *CISP '08: Proceedings of the 2008 Congress on Image and Signal Processing, Vol. 2*, p. 402–406, Washington, DC, USA. IEEE Computer Society.

[Ramakrishnan et al., 2005] Ramakrishnan, R., Goldstein, J., e Shaft, U. (2005). Similarity search in high-dimensional datasets. In *CVDB '05: Proceedings of the 2nd international workshop on Computer vision meets databases*, p. 1–2, New York, NY, USA. ACM.

[Ribeiro et al., 2009] Ribeiro, M. X., Bugatti, P. H., Traina, Jr., C., Marques, P. M. A., Rosa, N. A., e Traina, A. J. M. (2009). Supporting content-based image retrieval and computer-aided diagnosis systems with association rule-based techniques. *Data Knowl. Eng.*, 68(12):1370–1382.

[Roh et al., 2009] Roh, H., Kim, W.-C., Kim, S., e Park, S. (2009). A b-tree index extension to enhance response time and the life cycle of flash memory. *Inf. Sci.*, 179(18):3136–3161.

[Roweis & Saul, 2000] Roweis, S. T. e Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326.

[Rubner et al., 1998] Rubner, Y., Tomasi, C., e Guibas, L. J. (1998). A metric for distributions with applications to image databases. In *ICCV '98: Proceedings of the Sixth*

International Conference on Computer Vision, page 59, Washington, DC, USA. IEEE Computer Society.

[Russo et al., 2008] Russo, L. M. S., Navarro, G., e Oliveira, A. L. (2008). Fully-compressed suffix trees. In *LATIN'08: Proceedings of the 8th Latin American conference on Theoretical informatics*, p. 362–373, Berlin, Heidelberg. Springer-Verlag.

[Sakurai et al., 2002] Sakurai, Y., Yoshikawa, M., Uemura, S., e Kojima, H. (2002). Spatial indexing of high-dimensional data based on relative approximation. *The VLDB Journal*, 11(2):93–108.

[Santini & Jain, 1999] Santini, S. e Jain, R. (1999). Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):871–883.

[Santos et al., 2001] Santos, Roberto Figueira, F., Traina, A. J. M., Traina, Caetano, J., e Faloutsos, C. (2001). Similarity search without tears: The omni family of all-purpose access methods. In *Intl. Conf. on Data Engineering (ICDE)*, p. 623–630, Heidelberg, Germany. IEEE Computer Society.

[Schölkopf et al., 1998] Schölkopf, B., Smola, A., e Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319.

[Shasha & Wang, 1990] Shasha, D. e Wang, T. L. (1990). New techniques for best-match retrieval. *ACM Transactions on Information Systems (TOIS)*, 8(2):140–158.

[Skopal & Lokoč, 2008] Skopal, T. e Lokoč, J. (2008). Nm-tree: Flexible approximate similarity search in metric and non-metric spaces. In *DEXA '08: Proceedings of the 19th international conference on Database and Expert Systems Applications*, p. 312–325, Berlin, Heidelberg. Springer-Verlag.

[Smeulders et al., 2000] Smeulders, A., Worring, M., Santini, S., Gupta, A., e Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380.

[Sousa et al., 2007] Sousa, E. P., Traina, Jr., C., Traina, A. J., Wu, L., e Faloutsos, C. (2007). A fast and effective method to find correlations among attributes in databases. *Data Min. Knowl. Discov.*, 14(3):367–407.

[Stojmirović & Pestov, 2007] Stojmirović, A. e Pestov, V. (2007). Indexing schemes for similarity search in datasets of short protein fragments. *Inf. Syst.*, 32(8):1145–1165.

[Stricker & Orengo, 1995] Stricker, M. e Orengo, M. (1995). Similarity of color images. In Niblack, W. R. e Jain, R. C., editors, *Storage and Retrieval for Image and Video Databases III, Proc SPIE 2420)*, p. 381–392.

[Swain & Ballard, 1991] Swain, M. J. e Ballard, D. H. (1991). Color indexing. *IJCV: International Journal of Computer Vision*, 7(2):11–32.

[Tenenbaum et al., 2000] Tenenbaum, J. B., Silva, V., e Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

[Theodoridis et al., 2000] Theodoridis, Y., Stefanakis, E., e Sellis, T. K. (2000). Efficient cost models for spatial queries using r-trees. *IEEE Transactions on Knowledge and Data Engineering*, 12:19–32.

[Thomasian & Zhang, 2008] Thomasian, A. e Zhang, L. (2008). Persistent clustered main memory index for accelerating k-nn queries on high dimensional datasets. *Multimedia Tools Appl.*, 38(2):253–270.

[Traina & Jr, 2003] Traina, A. J. M. e Jr, C. T. (2003). Similarity search in multimedia databases. *Handbook of Video Databases - Design and Applications*, 1:711–738.

[Traina et al., 2002a] Traina, A. J. M., Jr., C. T., Bueno, J. M., e de A. Marques, P. M. (2002a). The metric histogram: A new and efficient approach for content-based image retrieval. In *Sixth IFIP Working Conference on Visual Database Systems*, p. 297–311, Brisbane, Australia.

[Traina et al., 2003a] Traina, A. J. M., Traina Jr., C., Bueno, J. M., Chino, F. J. T., e Azevedo-Marques, P. M. d. (2003a). Efficient content-based image retrieval through metric histograms. *World Wide Web Journal, Kluwer Academic Publishing Co.*, 6(2):157–185.

[Traina et al., 2003b] Traina, A. J. M., Traina Jr., C., Bueno, J. M., Chino, F. J. T., e Marques, P. M. d. A. (2003b). Efficient content-based image retrieval through metric histograms. *World Wide Web Journal*, 6(2):157–185.

[Traina et al., 2002b] Traina, Caetano, J., Traina, A. J. M., Santos, Roberto Figueira, F., e Faloutsos, C. (2002b). How to improve the pruning ability of dynamic metric access methods. In *Eleventh International Conference on Information and Knowledge Management (CIKM"2002)*, p. 219–226, McLean, VA - EUA. ACM Press.

[Traina et al., 2000] Traina, Caetano, J., Traina, A. J. M., Seeger, B., e Faloutsos, C. (2000). Slim-trees: High performance metric trees minimizing overlap between nodes. In Zaniolo, C., Lockemann, P. C., Scholl, M. H., e Grust, T., editors, *International Conference on Extending Database Technology*, v. 1777 of *Lecture Notes in Computer Science*, p. 51–65, Konstanz, Germany. Springer.

[Traina Jr et al., 1999] Traina Jr, C., Traina, A. J. M., e Faloutsos, C. (1999). Distance exponent : a new concept for selectivity estimation in metric trees. Research Paper CMU-CS-99-110, Carnegie Mellon University - School of Computer Science.

[Uhlmann, 1991] Uhlmann, J. K. (1991). Satisfying general proximity/similarity queries with metric trees. *Information Processing Letter*, 40(4):175–179.

[Valentine, 1975] Valentine, J. E. (1975). Angles in metric spaces. In Doe, A., editor, *Lecture Notes in Mathematics*, v. 490, p. 66–73. Springer Berlin / Heidelberg.

[Valle et al., 2008] Valle, E., Cord, M., e Philipp-Foliguet, S. (2008). High-dimensional descriptor indexing for large multimedia databases. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, p. 739–748, New York, NY, USA. ACM.

[Vespa et al., 2010] Vespa, T. G., Traina, Jr, C., e Traina, A. J. (2010). Efficient bulk-loading on dynamic metric access methods. *Inf. Syst.*, 35(5):557–569.

[Vieira et al., 2004] Vieira, M. R., Jr., C. T., Traina, A. J. M., e Chino, F. J. T. (2004). Dbm-tree: A dynamic metric access method sensitive to local density data. In *Brazilian Symposium on Databases (SBBD)*, Brasília, DF, Brazil.

[Volnyansky & Pestov, 2009] Volnyansky, I. e Pestov, V. (2009). Curse of dimensionality in pivot based indexes. In *SISAP '09: Proceedings of the 2009 Second International Workshop on Similarity Search and Applications*, p. 39–46, Washington, DC, USA. IEEE Computer Society.

[Wactlar et al., 1996] Wactlar, H. D., Kanade, T., Smith, M. A., e Stevens, S. M. (1996). Intelligent access to digital video: Informedia project. *IEEE Computer*, 29(5):46–52.

[Wang et al., 2010] Wang, J., Jiang, B., e He, Y. (2010). Shape-based search of mechanical cad models for product data management. *Int. J. Comput. Appl. Technol.*, 37(2):125–131.

[Wechsler et al., 2000] Wechsler, M., Munteanu, E., e Schäuble, P. (2000). New approaches to spoken document retrieval. *Inf. Retr.*, 3(3):173–188.

[White & Jain, 1996] White, D. A. e Jain, R. (1996). Similarity indexing with the ss-tree. *Data Engineering, International Conference on*, 0:516.

[Wilson & Martinez, 1997] Wilson, D. R. e Martinez, T. R. (1997). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34.

[Wilson, 1935] Wilson, W. A. (1935). On the imbedding of metric sets in euclidean space. *American Journal of Mathematics*, 57(2):322–326.

[Xiao et al., 2008] Xiao, Y.-y., Wang, X.-y., e Wang, F.-y. (2008). Shape-based similarity k nearest neighbor query for trajectory of moving objects. In *Mobility '08: Proceedings of the International Conference on Mobile Technology, Applications, and Systems*, p. 1–3, New York, NY, USA. ACM.

[Yang et al., 2010] Yang, L., Huang, X., Lv, R., Kang, M., e Yin, X. (2010). Performance of ss-tree with slim-down and reinsertion algorithm. In *ICMTMA '10: Proceedings of the 2010 International Conference on Measuring Technology and Mechatronics Automation*, p. 883–886, Washington, DC, USA. IEEE Computer Society.

[Yianilos, 1993] Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms - SODA*, p. 311–321, Austin, TX.

[Yoshihara et al., 2008] Yoshihara, T., Kobayashi, D., e Yokota, H. (2008). A concurrency control protocol for parallel b-tree structures without latch-coupling for explosively growing digital content. In *EDBT '08: Proceedings of the 11th international conference on Extending database technology*, p. 133–144, New York, NY, USA. ACM.

[Yu et al., 2007] Yu, B., Bailey, T., e Orlandic, R. (2007). Estimating the performance of multidimensional access methods based on nonoverlapping regions: Research articles. *Int. J. Intell. Syst.*, 22(3):259–277.

[Zhang & Zhang, 2008] Zhang, Z. e Zhang, R. (2008). *Multimedia Data Mining: A Systematic Introduction to Concepts and Theory*. Chapman & Hall/CRC.

[Zhou et al., 2007] Zhou, X., Zhou, X., e Shen, H. T. (2007). Efficient similarity search by summarization in large video database. In *ADC '07: Proceedings of the eighteenth conference on Australasian database*, p. 161–167, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

[Zhuang et al., 2008] Zhuang, Y., Zhuang, Y., Li, Q., Chen, L., e Yu, Y. (2008). Indexing high-dimensional data in dual distance spaces: a symmetrical encoding approach. In *EDBT '08: Proceedings of the 11th international conference on Extending database technology*, p. 241–251, New York, NY, USA. ACM.